Concept 2.6 User Manual

Volume 1

11/2007

Merlin Gerin
Square D
Telemecanique

Schneider Electric

Table of Contents

Safety Information XI

About the Book. XIII

Chapter 1 General description of Concept 1

At a Glance 1

1.1 General description of Concept. 3

At a Glance 3

Introduction 4

PLC hardware configuration 6

PLC Hardware Package Contents in Concept S, M and XL 7

1.2 Programming. 9

At a Glance 9

General information. 10

Libraries. 11

Editors 13

Online functions. 17

Communication 18

Secure Application 19

Utility program 21

Chapter 2 New Performance Attributes of Concept 2.6 in Comparison with Concept 2.5. 25

Introduction 25

New Performance Attributes of Concept 2.6 Compared with Concept 2.5 26

New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1 31

New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2 34

Chapter 3 **Project structure 35**
At a Glance 35
Project Structure and Processing 36
Programs 42
Sections 47
Configuration data 53

Chapter 4 **Creating a Project 55**
At a Glance 55
Overview 56
Step 1: Launching Concept 57
Step 2: Configuring the PLC 58
Step 2.1: Required Configuration. 59
Step 2.2: Optional Configuration 61
Step 3: Creating the User Program 64
Step 4: Save 66
Step 5: Perform Memory Prediction. 67
Step 6: Loading and Testing 68
Step 7: Optimize and Separate 73
Step 8: Documentation 75

Chapter 5 **PLC configuration 77**
At a Glance 77
5.1 General information about hardware configuration 79
At a Glance 79
General information 80
Proceed in the following way with the configuration 81
5.2 Configuration in OFFLINE and ONLINE mode 83
At a Glance 83
General information 84
Available Functions in OFFLINE and ONLINE Modes. 85
5.3 Unconditional Configuration. 87
At a Glance 87
Precondition. 88
PLC selection. 89
CPU Selection for the PLC Type 90
PLC memory mapping 95
Loadables 96
Segment manager 99
I/O Map 101

5.4 Optional configuration 105
At a Glance 105
Settings for ASCII Messages 106
Making Additional Functions Available in the Configurator 107
Data Exchange between Nodes on the Modbus Plus Network 108
How many words are really used when data is received (Peer Cop) 109
Protecting Data in the State RAM before Access 111
Parameterize interfaces 112
Special Options 114
5.5 Backplane Expander Config 117
At a glance 117
Generals to Backplane Expander 118
Edit I/O Map 119
Error handling 120
5.6 Configuration of various network systems. 121
At a Glance 121
Configure INTERBUS system 122
Configure Profibus DP System 123
Configure Ethernet 125
RTU extension. 127
Ethernet I/O Scanner. 128
How to use the Ethernet / I/O Scanner 131
5.7 Quantum Security Settings in the Configurator 133
Quantum Security Parameters 133

Chapter 6 Main structure of PLC Memory and optimization of memory 137
At a Glance 137
6.1 Main structure of the PLC Memory 139
General structure of the PLC Memory. 139
6.2 General Information on Memory Optimization. 141
Introduction 141
Possibilities for Memory Optimization 142
PLC-Independent 143
6.3 Memory Optimization for Quantum CPU X13 0X and 424 02 147
Introduction 147
General Information on Memory Optimization for Quantum CPU X13 0X and 424 02 148
Selecting Optimal EXEC File. 150
Using the Extended Memory (State RAM for 6x references) 154
Harmonizing the IEC Zone and LL984 Zone. 156
Harmonizing the Zones for Global Data and IEC Program Memory 158

6.4 Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B) 161
Introduction 161
General Information on Memory Optimization for
Quantum CPU 434 12(A) and 534 14(A/B) 162
Harmonizing IEC Zone and LL984 Zone 164
Harmonizing the Zones for Global Data and
IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)) 169
6.5 Memory optimization for Compact CPUs. 173
At a Glance 173
General Information on Memory Optimization for Compact CPUs 174
Harmonizing IEC Zone and LL984 Zone 176
Harmonizing the Zones for Global Data and
IEC Program Memory (Compact) 181
6.6 Memory optimization for Momentum CPUs 183
Introduction 183
General Information on Memory Optimization for Momentum CPUs. 184
Selecting Optimal EXEC file 186
Harmonizing the Zones for Global Data and
IEC Program Memory (Momentum). 187
6.7 Memory optimization for Atrium CPUs. 189
At a Glance 189
General Information on Memory Optimization for Atrium CPUs 190
Use of IEC 191
Harmonizing the Zones for Global Data and
IEC Program Memory (Atrium) 193

Chapter 7 Function Block language FBD 195
At a Glance 195
7.1 General information about FBD Function Block. 197
General information on Function Block language FBD 197
7.2 FBD Function Block objects. 199
At a Glance 199
Functions and Function Blocks (FFBs) 200
Link 204
Actual parameters 205
Text Object. 207
7.3 Working with the FBD Function Block langauge 209
At a Glance 209
Positioning Functions and Function Blocks 210
FFB Execution Order 212
Configuring Loops 214
7.4 Code generation with the FBD Function Block language 215
Code Generation Options 215

7.5 Online functions of the FBD Function Block language 217
Online Functions 217
7.6 Creating a program with the FBD Function Block language 219
Creating a Program in the FBD Function Block Language 219

Chapter 8 Ladder Diagram LD. 223
At a Glance 223
8.1 General information about Ladder Diagram LD 225
General Information about the LD Ladder Diagram Language 225
8.2 Objects in Ladder Diagram LD 227
At a Glance 227
Contacts 228
Coils 230
Functions and Function Blocks (FFBs) 233
Link 239
Actual Parameters. 240
Text object. 242
8.3 Working with the LD Ladder Diagram 243
At a Glance 243
Positioning Coils, Contacts, Functions and Function Blocks. 244
Execution sequence 246
Configuring Loops 248
8.4 Code generation with LD Ladder Diagram 249
Code Generation Options 249
8.5 Online functions with the LD Ladder Diagram. 251
Online Functions 251
8.6 Creating a program withLD Ladder Diagram. 253
Creating a Program in LD 253

Index i

The chapters marked gray are not included in this volume.

Chapter 9 Sequence language SFC 257

Chapter 10 Instruction list IL. 307

Chapter 11 Structured text ST. 377

Chapter 12 Ladder Logic 984 439

Chapter 13 DFBs (Derived Function Blocks). 469

Chapter 14 Macros 511

Chapter 15 Variables editor. 535

Chapter 16 Project Browser 549
Chapter 17 Derived data types 557
Chapter 18 Reference data editor 587
Chapter 19 ASCII Message Editor 605
Chapter 20 Online functions. 627
Chapter 21 Import/Export 689
Chapter 22 Documentation and Archiving 733
Chapter 23 Simulating a PLC 751
Chapter 24 Concept Security 763
Appendices 779
Appendix A Tables of PLC-dependent Performance Attributes. 781
Appendix B Windows interface 805
Appendix C List of symbols and short cut keys 825
Appendix D IEC conformity 857
Appendix E Configuration examples 887
Appendix F Convert Projects/DFBs/Macros 1001
Appendix G Concept ModConnect 1005
Appendix H Convertion of Modsoft Programs. 1013
Appendix I Modsoft and 984 References 1019
Appendix J Presettings when using Modbus Plus for startup 1023
Appendix K Presettings when using Modbus for startup. 1037
Appendix L Startup when using Modbus with the EXECLoader 1043
Appendix M Startup when using Modbus with DOS Loader. 1059
Appendix N Startup when using Modbus Plus
with the EXECLoader. 1073
Appendix O Startup when using Modbus Plus
with DOS Loader 1093

Appendix P EXEC files 1109
Appendix Q INI Files 1115
Appendix R Interrupt Processing. 1135
Appendix S Automatic Connection to the PLC 1159
Glossary 1169

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡ The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

⚠ This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

About the Book

At a Glance

Document Scope

This user manual is intended to help you create a user program with Concept. It provides authoritative information on the individual program languages and on hardware configuration.

Validity Note

The documentation applies to Concept 2.6 for Microsoft Windows 98, Microsoft Windows 2000, Microsoft Windows XP and Microsoft Windows NT 4.x.

Note: Additional up-to-date tips can be found in the Concept README file.

Related Documents

Title of Documentation	Reference Number
Concept Installation Instructions	840 USE 502 00
Concept IEC Block Library	840 USE 504 00
Concept EFB User Manual	840 USE 505 00
Concept LL984 Block Library	840 USE 506 00

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

General description of Concept

1

At a Glance

Overview

This chapter contains a general description of Concept. It should provide an initial overview of Concept and its helper programs.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	General description of Concept	3
1.2	Programming	9

1.1 General description of Concept

At a Glance

Overview

This section describes the performance features of Concept and provides an overview of the hardware that may be programmed using Concept.

What's in this Section?

This section contains the following topics:

Topic	Page
Introduction	4
PLC hardware configuration	6
PLC Hardware Package Contents in Concept S, M and XL	7

Introduction

Operating System

Nowadays, a graphical user interface is a requirement for tasks of this kind. For this reason, Concept has been established as an MS Windows application. Concept can be operated in Windows 98, Windows 2000, Windows XP and Windows NT. These operating systems have the advantage that they are used all over the world. Therefore PC users have a basic knowledge of Windows technology and mouse operation. In addition to this all common monitors, graphic cards and printers can be used with MS Windows. As a user, you are not therefore tied to specific hardware configurations.

International Standard IEC 1131-3

For effective system configuration Concept offers a unified configuration environment in accordance with international standard regulations IEC 1131-3.

PLC Independence when Programming

The guiding principle behind the development of Concept was that all the system configuration procedures and all the editors should have the same look and feel. Most of the configuration steps, especially program creation, are designed independently of the PLC to be programmed.

Graphical Interface

The entire program is divided up into sections corresponding to the logic structure.

The Concept configuration tool enables objects (such as function blocks, steps, and transitions) to be selected, placed and moved easily in graphical form. Plausibility tests already take place in the SFC editor (Sequential Function Chart/ sequence language) during object placing, as most of the links between objects are generated automatically during placing. In the FBD editor (Function Block Diagram/Function Block language) and LD editor (Ladder Diagram), plausibility tests take place when blocks are linked. Unauthorized links, such as those between different data types have already been rejected during configuration. A plausibility test also takes place in the LL984 editor (Ladder Logic 984) during placing. In the IL editor (Instruction List) and ST editor (Structured Text) unauthorized instructions are identified via a colored outline. After the first successful program run, the program may be optimized in graphic terms by moving links, blocks or texts to improve the display.

Print

If desired the sections may be displayed with print preview information, in order to individually control pages of documentation. Signals receive an expansive designation with symbol names and comments. Unique notes on signal tracking are provided at the signal breaks. The individual block processing sequences from one section may be displayed and documented in the FBD editor.

Import/Export Functions

Sections from various projects can be combined as desired in another project using import/export functions.

It is also possible to convert the sections of one IEC programmer language into sections of another IEC programmer language.

Variables may be imported into and exported from the text using text delimited or FactoryLink format.

Runtime System

The runtime system on the PLC offers quick reactions to signal state process changes (short cycle time), Simulating signal transmitters (see *Simulating a PLC, p. 751*), Online display (see *Online functions, p. 627*), online parameter changes and online program changes.

Open Software Architecture

Concept possesses open software architecture to enable connection to external systems (e.g. for visualization) via standard interfaces.

Online Help

Special care was taken when developing the help function. The context sensitive Online help function (see *How the Online Help is set out, p. 821*) provides support for every configuration situation just by clicking on the subject using the mouse or pressing the F1 key. Menu commands and dialogs are also context sensitive, as are, function blocks and hardware components of the individual PLC families.

PLC hardware configuration

Description

Concept is the unified projection tool for Quantum, Compact, Momentum and Atrium products.

Hardware components (for example CPU, program memory, input/output units etc.) can be specified before, during or after program creation.

This projection task can be performed both online (linked to the PLC) and locally (PC alone). Projection is supported by Concept, and only suggests valid combinations. Misprojection is therefore prevented. In online mode the projected hardware is tested for plausibility immediately and input errors are rejected.

After linking the programmer device (PC) to the PLC, a plausibility test is performed on the projected values (e.g. from the Variable Editor) using the actual hardware resources and if necessary an error message will appear.

PLC Hardware Package Contents in Concept S, M and XL

Description

PLC Hardware Package Contents in Concept S, M and XL:

Concept version	contain Hardware
Concept Vx.x **S**	Momentum
Concept Vx.x **M**	Compact, Momentum
Concept Vx.x **XL**	Atrium, Compact, Momentum, Quantum

1.2 Programming

At a Glance

Overview

This section provides an overview of the editors which are available in Concept.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	10
Libraries	11
Editors	13
Online functions	17
Communication	18
Secure Application	19
Utility program	21

General information

At a Glance

As a solution for automatic control engineering tasks, Concept provides the following IEC 1131-3 compatible programming languages:
- Function Block language FBD (Function Block Diagram) (see *FBD editor, p. 13*),
- LD (Ladder Diagram) (see *LD editor, p. 14*),
- Sequential language SFC (Sequential Function Chart) (see *SFC editor, p. 14*),
- Instruction List IL (see *IL editor, p. 15*) and
- Structured Text ST (see *ST editor, p. 15*).

The Modsoft orientated language is also available
- Ladder Diagram LL984 (Ladder Logic) (see *LL984 editor, p. 16*).

The IEC programming language (FBD, LD, SFC, ST and IL) basic elements are Functions and Function Blocks, which make up assembled logic units. Concept contains various Block libraries (see *Libraries, p. 11*) with predefined elementary functions/Function Blocks (EFBs). In order to locate the individual EFBs without difficulty, they are split into different groups according to their area of use.

For the Modsoft orientated programming language LL984, there is a Block library (see *Libraries, p. 11*) with Instructions available.

Sections

The control program is constructed from sections according to the logic structure. Only one programming language is used within a section.

Merging these sections makes up the entire control program and the automation device uses this to control the process. Any IEC sections (FBD, LD, SFC, IL, ST) may be mixed within the program. The LL984 sections are always edited as a block before the IEC sections.

Data types

A subset of Data types from the international standard IEC1131-3 is available.

In the Data type editor (see *Data type editor (DDT editor), p. 16*) intrinsic data types can be derived from IEC data types.

Using variables

Variables for linking basic elements (objects) within a section are not usually necessary with the graphic programming languages FBD, LD, SFC and LL984, as these links are usually made graphically. (An additional link using variables is only necessary for incredibly complex sections.) Graphic links are managed by the system and therefore no projection requirement is created. The Variable Editor (see *Variable Editor, p. 16*) is used to project all other variables such as those for data transfer between various sections.

Libraries

At a Glance

For program creation Concept provides various block libraries with predefined Functions and Function Blocks.

There are 2 different types of block libraries:

- IEC library
  Block libraries for sections in the IEC programming languages (FBD, LD, SFC, IL and ST)
- LL984 Library
  Block library for sections in the Modsoft orientated programming language LL984

IEC library

The following IEC libraries are available for applications:

- **AKFEFB**
  This library contains the AKF/ALD EFBs, which are not covered by the IEC library.
- **ANA_IO**
  This library is for analog value processing.
- **COMM**
  This library is used for exchanging data between a PLC and another Modbus, Modbus Plus or Ethernet node.
- **CONT_CTL**
  This library is for projecting process-engineering servoloops. It contains controller, differential, integral, and polygon graph EFBs.
- **DIAGNOSTICS**
  This library is used to investigate the control program for misbehaviors. It contains action diagnostics, Reaction diagnostics, locking diagnostics, process prerequisite diagnostics, dynamic diagnostics and signal group monitoring EFBs.
- **EXPERTS**
  This library contains EFBs, which are necessary for using expert modules.
- **EXTENDED**
  This library contains useful supplements for different libraries. It has EFBs for creating average values, selecting maximum values, negating, triggering, converting, creating a polygon with 1st degree interpolation, edge recognizing, and for specifying an insensitive zone for control variables.
- **FUZZY**
  This library contains EFBs for fuzzy logic.
- **IEC**
  This library contains the EFBs defined in IEC 1131-3. It has for example EFBs for mathematical calculations, counters, timers etc.
- **LIB984**
  This library contains IEC 1131 compatible EFBs from the LL984 library, for example, EFBs for register transfer.
- **SYSTEM**
  This library contains EFBs for using system functions. It has EFBs for cycle time recognition, for various system cycle use, for SFC section control and for system status display.

LL984 Library

The LL984 library contains the LL984 editor instructions (blocks). It contains instructions for mathematical calculations, counters, timers, instructions for displaying system status, controller, differential and integral instructions and instructions for exchanging data between a PLC and another Modbus or Modbus Plus node.

Editors

At a Glance

When generating a section specify which programming language you are going to use.

The following editors are available for creating sections in the various programming languages:

- FBD editor (Function Block Language) (see *FBD editor, p. 13*)
- LD editor (Ladder Diagram) (see *LD editor, p. 14*)
- SFC editor (Sequence language) (see *SFC editor, p. 14*)
- IL editor (Instruction List) (see *IL editor, p. 15*)
- ST editor (Structured Text) (see *ST editor, p. 15*)
- LL984 editor (Modsoft orientated Ladder Logic) (see *LL984 editor, p. 16*)

The following editors are available for declaring variables, creating data types and displaying variables.

- the Variable Editor (for declaring variables), (see *Variable Editor, p. 16*)
- the reference data editor (for displaying and online changing of values) (see *Reference data editor, p. 16*) and
- the data type editor (for creating user specific data types) (see *Data type editor (DDT editor), p. 16*).

The following editors are available for creating user specific functions and Function Blocks:

- Concept DFB (for creating Derived Function Blocks and macros) (see *Concept DFB, p. 22*)
- Concept EFB (for creating user specific elementary functions and Function Blocks) (see *Concept EFB, p. 23*)

FBD editor

The FBD editor (see *Function Block language FBD, p. 195*) is used for graphic function plan programming according to IEC 1131-3.

Elementary functions, Elementary Function Blocks (EFBs) and Derived Function Blocks (DFBs) are connected with signals (variables) onto FBD sections for the function plan. The size of a FBD section is 23 lines and 30 columns.

EFBs are equipped with a fixed or variable number of input variables and may be placed anywhere on the section. Variables and EFBs may have comments separately added to them, column layouts on a section may be commented on anywhere using text boxes. All EFBs may be performed conditionally or unconditionally.

All the EFBs are divided into function- and use-orientated libraries in various groups, to make them easier to locate.

LD editor

The LD editor (see *Ladder Diagram LD, p. 223*) is used for graphic ladder programming according to IEC 1131-3.

Contacts and coils are connected to the Ladder Diagram in LD sections using signals (variables).

The size of a FBD section is 23 lines and 30 columns.

Furthermore, the elementary functions and Function Blocks (EFBs), which are named in the FBD editor, the Derived Function Blocks (DFBs) and User Defined Function Blocks (UDFBs) may also be bound in the ladder diagram (see *FBD editor, p. 13*).

The structure of a LD section corresponds to a rung for relay switching. The left power rail is located on its left-hand side. This left power rail corresponds to the phase (L ladder) of a rung. With LD programming, in the same way as in a rung, only the LD objects (contacts, coils) which are linked to a power supply, that is to say connected with the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder, is not shown optically. However, all coils and EFB outputs are linked with it internally and this creates a power flow.

SFC editor

The SFC editor (see *Sequence language SFC, p. 257*) is used to graphically program an IEC 1131-3 compatible sequential control.

The SFC elements are connected in a SFC section to one of the sequential controls corresponding to the task setting. The size of a SFC section is 32 lines and 200 lines.

The following sequential control programming objects are available in Concept.
- Step (including actions and action sections)
- Transition (including transition section)
- Alternative branch and merge
- Parallel branch and merge
- Jump
- Connection

Simple diagnostics monitoring functions are already integrated in the steps.

IL editor

The IL editor (see *Instruction list IL, p. 307*) is used for programming IEC 1131-3 compatible instruction lists.

Existing IL instructions, elementary functions and Elementary Function Blocks (EFBs), and Derived Function Blocks (DFBs) are written in series in text form in IL sections from operators (commands) and operands (signals, variables).

When the program is entered, all the standard Windows services and some additional commands for text-processing are available. The size of an IL section is 64 Kbyte maximum.

The following instruction list programming operators are available in Concept:

- Logic (AND, OR etc.)
- Arithmetic (ADD, SUB, MUL, DIV, …)
- Comparative (EQ, GT, LT, …)
- Jumps (JMP, … conditional/unconditional)
- EFB call (CAL , … conditional/unconditional)

IL programming is done in text form. When text is entered, all the standard Windows services for text-processing are available. The IL editor also contains some further commands for text-processing.

A spell check is performed immediately after text has been entered (instructions, key words, separators), highlighting errors with a colored outline.

ST editor

The ST editor (see *Structured text ST, p. 377*) is used for programming IEC 1131-3 structured text.

Existing ST statements, elementary functions and Elementary Function Blocks (EFBs), and Derived Function Blocks (DFBs) are written in text form in IL sections by printing (operator lists) and operands (signals, variables).

When the program is entered, all the standard Windows services and some additional commands for text-processing are available. The size of a ST section is 64 Kbyte maximum.

The following structured text programming statements and operators are available in Concept:

- conditional/unconditional statement execution (IF, ELSIF, ELSE, …)
- conditional/unconditional loop execution (WHILE, REPEAT)
- Mathematical, comparative, and logic operators
- conditional/unconditional EFB call

ST programming is done in text form. When text is entered, all the standard Windows services for text-processing are available. The ST editor also contains some further commands for text-processing.

A spell check is performed immediately after text has been entered (instructions, key words, separators), highlighting errors with a colored outline.

LL984 editor

Using the Modsoft orientated LL984-Editor (see *Ladder Logic 984, p. 439*) (Ladder Diagram 984), instructions, contacts, coils and signals (variables) are connected to a ladder diagram. Instructions, contacts, coils and variables may be commented on.

The structure of a LL984 section corresponds to a rung for relay switching. The left power rail is located on its left-hand side, but it is not visually displayed. This left power rail corresponds to the phase (L ladder) of a rung. With LL984 programming, in the same way as in a rung, only the LL984 objects (instructions, contacts, coils) connected to a power supply, i.e. connected to the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder is not visually displayed either. However, all coils and instruction outputs are linked with it internally and this creates a power flow.

Concept has various predefined instructions for ladder programming using LL984. These may be found in the block library LL984. Additional instructions for special applications are available as loadables and may be loaded at a later time.

Variable Editor

The Variable Editor (see *Variables editor, p. 535*) is used to declare and comment on all necessary symbolic signal names (variables). Only declared variables may be used in Concept programs.

A data type must be assigned to each symbolic signal name! If this variable is assigned a reference address, a Located variable (without reference address = Unlocated variable) is received. An initial value may also be provided for each variable, which will be transferred into the PLC during the first load.

Data type editor (DDT editor)

The Data type editor (see *Derived data types, p. 557*) may be used to define specific Derived Data Types (Derived Data Type = DDT).

Derived Data Types combine several Elementary data types (BOOL, WORD, …) in one data record. It is not only the same data types which may be combined as ARRAY, but also various data types may be combined as STRUCT. In Concept, a number of Derived Data Types are already available, which for instance may be used for DFBs.

DDTs appear in DFBs or EFBs only as a connection, i.e. for instance in FBD a variable input is only necessary in the block. It is thus recommended that frequently recurring groups of elementary data types (and also DDTs) be defined as DDTs, in order to improve accessibility of an application.

The definition appears in text form, and all the standard Windows services and some additional commands for text-processing are available. The size of a data type file is 64 Kbyte maximum.

Reference data editor

The Reference data editor (see *Reference data editor, p. 587*) may be used in online mode to display the variable value, to force variables and to set variables. There is also the possibility of separating variables from the process. Inputs may be saved in a data file and be reused.

Online functions

Available online functions

After the programming device has been linked to the PLC, a range of online Startup and maintenance functions become available.

- the program on the programming device is compared with the program on the PLC
- the PLC can be started and stopped
- Object information is displayed
- Programs can be loaded, sections can be changed online and loaded
- Variable values can be entered online
- Animation mode shows the program with its current signal states

Operating and monitoring

Declaration of special operating and monitoring variables is not necessary in Concept. The variables to be visualized can be identified as such in the Variable Editor and then be exported into a ModLink or FactoryLink configuration data file. This data file can be used for visualizing.

Communication

Description

Communication between the PLC and another Modbus-, Modbus Plus-, SY/MAX-Ethernet or TCIP/IP Ethernet node is projected using IEC languages (FBD, LD, SFC, ST, IL) with the EFBs from the block library COMM. The instruction MSTR may be used with the programming language LL984 to construct these communications.

A peer to peer transfer of register contents is possible using the peer cop, independent of these blocks/instructions.

Communication is projected between the PLC and the decentralized I/O via the INTERBUS by simply entering the NOA module in the component list and loading a loadable (ULEX).

Communication is projected between the programming device and a PLC via Ethernet by simply entering and parametering the appropriate couple module in the component list.

Secure Application

At a Glance

In several areas of industry, the need for security demands regulated access to PLCs, recording program changes and archiving those recordings. Following a standardized procedure ensure that records may not be falsified. To enable these requirements, new features have been implemented in Concept that ensure secure application. To guarantee that all of these parameters are defined, the user can activate the **Secure Application** check box in the **Project** → **Project Properties** dialog. Concept will then ensure that all of these parameters are set and that their contents remain valid. The project is then indicated as being a secure application, and this information is included in the information that is downloaded to the PLC.

Secure Application

The secure application is defined in the **Project** → **Project Properties** dialog by activating the **Secure Application** check box. These settings are then exported, imported, read and loaded to the PLC.

Note: When the secure application is activated, a NOT EQUAL status is generated and required reloading to the PLC. Unchecking the check box also creates a NOT EQUAL status so that loading is again required as well. If Concept is connected to a PLC that is already defined with the "Secure Application" setting, the setting is automatically accepted in Concept in case of upload the controller.

The log file is stored in the Concept directory and has the name of the current date (YEARMONTHDAY.ENC, e.g. 20020723.ENC). The path of the log file can be defined in dialog **Common Preferences**. If no path is defined then Concept uses the default log path (Concept directory, e.g. C:\CONCEPT).

Among other things, logging write-access to the PLC can record the following data:

- Section name
- EFB/DFB Instance name, FB Type name
- Pin Name
- [Variable name] [Literal] [Address]
- Old value
- New value
- User name (if the Concept (Login) password is activated in Concept Security)
- Data and Time (see also*Address format in LOG file [Logging], p. 1127*)

Requirements

The secure application can only be activated if the following prerequisites are met:

- can only be used with 140 CPU 434 12A or 140 CPU 534 14A/B
- at least one IEC section (if no IEC section exists then the download is aborted.)
- Offline mode (**Online** → **Disconnect...**)
- Supervisor Rights (see Concept under **Help** → **About...** → **Current User:**)

Activation Combination for Secure Application

Various Activation Combinations for Secure Application:

"Secure Application" activated in Concept	"Secure Application" loaded to PLC	Reaction to connection with the PLC
Not activated	Not activated	Normal operation without secure application
Not activated	Activated	When uploading, the **Secure Application** check box is activated in Concept and encrypted logging is activated.
Activated	Not activated	Download required because the status is NOT EQUAL.
Activated	Activated	Normal operation with secure application (e.g. encrypted logging).

Reading the Encrypted Log File

To read the encrypted log file, the View tool is opened automatically in the **View Logfile** dialog.

Note: If an encrypted log file has been improperly modified in any way, the log is decoded as much as is possible, and the lines that have been modified will remain unreadable. The first line will contain the message: "This log file has been modified".

Utility program

At a Glance

In addition to Concept the following range of utility programs are available:

- Concept DFB
- Concept EFB
- Concept SIM (16 bit)
- Concept PLCSIM32 (32 bit)
- Concept Security
- Concept WinLoader
- Concept Converter
- Concept ModConnect

Concept DFB

Concept DFB is used to create DFBs (Derived Function Blocks) (see *DFBs (Derived Function Blocks), p. 469*) and Macros (see *Macros, p. 511*).

DFBs (Derived Function Blocks)

DFBs can be used for setting both the structure and the hierarchy of a program. In programming terms, a DFB represents a subroutine.

DFBs can be created in the programming languages FBD, LD, IL, and ST. In Concept, DFBs can be called up in any programming language, regardless of the programming language they were created in. One or several existing DFBs can be called up within one DFB, with the called-up DFBs themselves able to call up one or several DFBs.

Macros

Macros are used to duplicate frequently used sections and networks (including their logic, variables and variable declaration).

Macros have the following properties:
- Macros can only be created in the programming language FBD.
- Macros only contain one section.
- Macros can contain a section of any complexity.
- In programming terms, there is no difference between an instanced macro, i.e. a macro inserted into a section and a conventionally created section.
- It is possible to call up DFBs in a macro.
- It is possible to declare macro-specific variables for the macro.
- It is possible to use data structures specific to the macro
- Automatic transfer of the variables declared in the macro.
- Initial values are possible for the macro variables.
- It is possible to instance a macro many times in the entire program with different variables.
- Section names, variable names and data structure names can contain the character ~ as an exchange marking.

Concept EFB

The optional tool Concept EFB can be used to generate, in C++ programming language, your own application specific Functions and Function Blocks (EFBs) and to integrate them in the form of libraries with groups in your version of Concept.

The operating rules for these user-defined blocks (UDFBs) are identical to those for standard EFBs.

It is, for instance, recommended that complex program parts with a high number of calls and program parts, whose solution is to remain hidden from the user, e.g. special technology objects etc. be generated using Concept EFB.

Note: Concept EFB is not included as part of the Concept package and may be ordered in addition.

Concept SIM (16 bit)

The 16 bit simulator Concept SIM (see *Simulating a PLC (16-bit simulator), p. 753*) is available for simulating a PLC, i.e. to test your user program online without hardware. Concept SIM simulates a coupled PLC via Modbus Plus.

Note: The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

Concept PLCSIM (32 bit)

The 32 bit simulator Concept PLCSIM32 (see *Simulating a PLC (32-bit simulator), p. 755*) is available for simulating a PLC, i.e. to test your user program online without hardware. Concept PLCSIM32 simulates a PLC coupled via TCP/IP, where the signal states of the I/O modules can also be simulated. Up to 5 programming devices can be coupled to the simulated PLC at the same time.

Note: The simulator is only available for the IEC languages (FBD, SFC, LD, IL and ST).

Concept Security

Concept Security (see *Concept Security, p. 763*) can be used to assign access. Access signifies that the function of Concept and its utility programs is limited depending on the user.

The access defined for one user is applicable to all Concept installation projects. A maximum of 128 users may be defined.

Concept Converter

Projects, DFBs, macros, and data structures (Derived Data Types), created for an earlier version of Concept, can be converted without hassle to work in the current version of concept in the Concept Converter (see *Convert Projects/DFBs/Macros, p. 1001*).

Concept EXECLoader

The Concept EXECLoader can be used to load Exec data files onto the PLC.

Concept ModConnect

Concept-ModConnect (see *Concept ModConnect, p. 1005*) can be used to extend the configurator for new (specific) I/O modules.

New Performance Attributes of Concept 2.6 in Comparison with Concept 2.5

2

Introduction

Overview

This Chapter describes the new performance attributes of Concept 2.6 in comparison with Concept 2.5.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
New Performance Attributes of Concept 2.6 Compared with Concept 2.5	26
New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1	31
New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2	34

New Performance Attributes of Concept 2.6 Compared with Concept 2.5

Highlights

New general performance attributes:

- **Interrupt sections**
- **Global variables**
- **Security features**

New EFBs

New EFBs in the SYSTEM library:

New EFBs	Description
I_LOCK	Disable all interrupt sections
I_UNLOCK	Enable all interrupt sections
I_MOVE	Interrupt protected assignment
ISECT_OFF	Disable specific interrupt sections
ISECT_ON	Unlock a specific interrupt section
ISECT_STAT	Interrupt section status
PRJ_VERS	States project name and version
GET_IEC_INF	Read IEC status flags
RES_IEC_INF	Reset IEC status flags

New EFBs in the COMM library:

New EFBs	Description
PORTSTAT	States Modbus Port status

Start Concept

New features when starting Concept:

New performance attributes	Description
Automatic connection to every **desired** PLC	Startup using the Concept Project Symbol creates automatic connection to any desired PLC. This connection is defined by the Command line parameter (see *Automatic Connection with Command Line Parameters (Modbus, Modbus +, TCP/IP), p. 1160*).
When starting Concept using the CCLaunch tool, a connection is made to every **desired** PLC	In large networks, a topology file is created and is then used in the CCLaunch tool. You can use this to create a complete MB+ Routing path (see *Automatic Connection with the CCLaunch Tool (Modbus Plus), p. 1163*), which then creates a connection to the PLC automatically.
Displays list of previously opened Projects/ DFBs	When starting Concept a list of previously opened Projects/DFBs (max. 4) is displayed in the **File** main menu.
Archive content display	When unpacking an archived project, all archived files are shown first.

Animation

12 different color schemes for animation in the FBD, IL, ST, SFC and LD editors:

New performance attributes	Description

CONCEPT.INI:
`[Colors]`
`AnimationColors= (0-12)`
	Defines the color scheme for online animation in all editors.

Reference data editor

New feature in the reference data editor:

New performance attributes	Description
Address format IEC (QW0000X)	The IEC (QW0000X) address format can be displayed.

Online functions

New online features:

New performance attributes	Description
Quantum password protection	Quantum PLC is write protected by entering a password.
Event sections	Online diagnostics are displayed for Interrupt sections.
Event viewer	Error descriptions can be defined in a project specific INI file (see *INI Settings for the Event Viewer [Online Events], p. 1133*) that should appear in the event viewer (**Online** → **Online events...**).

Message window

New performance attributes in the Windows menu:

New performance attributes	Description
Save messages	After messages are displayed they can be saved to file using the **Save Messages...** (main menu **Window**) menu command.

New CPU

New CPU:

PLC family	Description
Atrium	CPU 180-CCO-241-11

New Module

New Quantum module:

Module	Description
140-NOE-771-01	Ethernet module without Hot Standby features.
140-NOE-771-11	Ethernet module (Factory Cast) without Hot Standby features.
140-CPS-114-20	Power supply module
140-CPS-124-20	Power supply module
140-NOG-111-00	1/SFB Master module
140-NWM-100 00	Ethernet module (Factory Cast HMI)

New Momentum module:

Module	Description
170-ANR-120-91	Analog/Digital Input/Output module

Project Browser

New features in the Project browser:

New performance attributes	Description
Display interrupt sections	When I/O event sections and Timer event sections are used, they are displayed in the Project browser structure.
Show detailed view	The Project browser window is split vertically, and a second window displays the substructure (e.g. DFBs, Transitions sections, etc.) of the selected elements in a structure tree.

Analyze section

New features when analyzing sections:

New performance attributes	Description
Analyze interrupt sections	There is now an additional analysis for Interrupt sections.
Analyzing global variables in DFBs	There is an analysis for global variables in DFBs.

DFB

New features for DFB programming:

New performance attributes	Description
Located variables	Located variables are permitted in DFBs when the option in the **IEC Extensions** dialog box is enabled. Global variables can be created throughout the program with located variables in DFBs.

Data types

New features for DFB programming:

New performance attributes	Description
View comments for data structure elements	Comments for data type components defined in data type files (*.ddt, *.dty) are displayed in: ● Editors status line ● Variables editor for the definition of initial values ● Inspect Animation field
Extended Data Type Definition (larger than 64 Kbytes), p. 565	The 64 kb restriction is not imposed for local data type definition with the introduction of unlocated Include files.

Configuration

New features in the Configurator:

New performance attributes	Description
1/SFB Coupler configuration	Required to provide support for the A500/A350 I/O module. Extended I/O range up to 160 input/output words.
Quantum security parameter	The following parameters can be defined in the new dialog box (submenu of the **Config. Extensions**): ● Secure data area ● Network write restrictions ● Enable the Auto-Logout option
Interbus configuration with Atrium	The Interbus configuration is done with Atrium CPUs 180 CCO 241 01 (= 1 INTERBUS) and 180 CCO 241 11 (= 2 INTERBUS).

Logging (*.LOG, *.ENC)

New features for DFB logging:

New performance attributes	Description
Additional contents	When logging PLC write access, modifications made to variable and literal values are displayed in addition.
New Date/Time format	By activating the check box **Universal Date Format** in dialog **Common Preferences** (setting also affects the CONCEPT.INI file) the format can be changed. The month is then stated within Concept with 3 characters and in English. **Example:** 24-Dec-2002 14:46:24
Encrypting the log	By activating the check box **Encrypt Logfile** in dialog **Common Preferences** (or indirectly using the check box **Secure Application** in dialog **Project Properties**) login the write access to the PLC will be encrypted. The encrypted file contains the file extension *.ENC.

Secure Application

New features for a secured application:

New performance attributes	Description
Application backup	If you activate the check box in the **Project** → **Project Properties** dialog box, program modifications are automatically logged and encrypted in a *.ENC file. These settings can be loaded using Export/Import and transferred to the PLC.

New Tools

New Tools for Concept:

New Tool	Description
CCLaunch	This tool is used for making an automatic connection (see *Automatic Connection with the CCLaunch Tool (Modbus Plus), p. 1163*) with a PLC in a large network.
View Tool	This tool allows you to view encoded LOG files (*.ENC). It is started automatically with menu instruction **View Logfile** if log encrypting has been activated.

New performance attributes of Concept 2.6 SR2 in comparison with Concept 2.6 SR1

New EFBs

New EFBs in the IEC library:

New EFBs	Description
CMPR	Compares the Bit pattern of Matrix A to that of Matrix B.
MBIT with pointer	Changes the bit position in a data matrix.
SEARCH	Searches the register in a source table for a specific bit pattern.
SENS with pointer	Checks the query value of a specific bit position in a data matrix.
XXOR	Performs a Boolean Exclusive-OR-Operation with the bit patterns of the source and target matrix.

Search/ Replacement of FFBs

New features when searching for/replacing FFBs:

New feature	Description
FFB type is replaced in all sections (only for DFBs)	In the dialog box **Replace FFB Type** by activating the new check box **Replace in all sections** the selected FFB type can be replaced in all sections (only for DFBs).

Create a new project

New features when generating a new project:

New feature	Description
Specify project path when generating a new project	When generating a new project (**File** → **New Project**) you can define a new path or accept the standard path again.

New options in the upload and loading dialog box

New options in the upload and loading dialog box:

New features	Description
New check boxes in the dialog box **Load into the PLC**: - State RAM + Initial Values - Only state RAM	By activating the check box **State RAM + Initial Values** at first all initial values of the Located 4x-Variables are copied from the Variable Editor into the state RAM mirror. Then, the initial values and all blocked 0x and 1x-I/O-bits are loaded from the state RAM mirror into the PLC. By activating the check box **State RAM Only** the initial values of the Located 4x-Variables and all blocked 0x and 1x I/O bits are loaded from the state RAM mirror into the PLC.
New check boxes in the dialog box **PLC Upload**: - Upload State RAM + Initial Values - Only upload State RAM	By activating the check box **Upload State RAM + Initial Values** at first all Located 0x-, 1x, and 4x-values are read from the PLC and saved in the state RAM mirror. Then, the initial values of the 4x-variables are overwritten with the value from the state RAM mirror. With the activation of the check box **Only read state RAM** all Located 0x-, 1x- and 4x-values are read from the SPS, and saved in the state RAM mirror.

INI files

New settings in the CONCEPT.INI:

New Settings	Description
Define overwriting of the uploaded state RAM values	In the line [RDE] of the CONCEPT.INI you can define that uploaded state RAM values are not overwritten by online operations in the RDE.
Define start of the RDE-Animation	In the line [RDE] of the CONCEPT.INI you can define that the RDE animation is automatically started when opening a table.
Exclusion of all or global DFBs from Online-Backup	In the line [Backup] of the CONCEPT.INI you can define that after the Online-Backup the directories "DFB" and/or "DFB.GLB" are not present in the backup directory.

New settings in the Projectname.INI:

New Setting	Description
Define path and backup files	In the line [Backup] of Projectname.INI, you can output a Batch-file (EXE-file) for the Online-Backup-Operation, by which you perform additional backups e.g. for another PC.

Multiple Address Assignment

New feature for multiple address assignment:

New feature	Description
Cleaning up multiple assignment of a single address by different variable names	In the dialog box **Multiple Address Assignments** variable names that are all assigned to the same address are replaced or renamed. In the end, only one variable name is assigned to this address.

New performance attributes of Concept 2.6 SR3 in comparison with Concept 2.6 SR2

New menu command

New menu command:

New menu command	Description
Options → Tools	Use this menu command to open a menu to execute additional applications or help programs.

Project structure

3

At a Glance

Overview

This chapter describes the structure of projects in Concept.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Project Structure and Processing	36
Programs	42
Sections	47
Configuration data	53

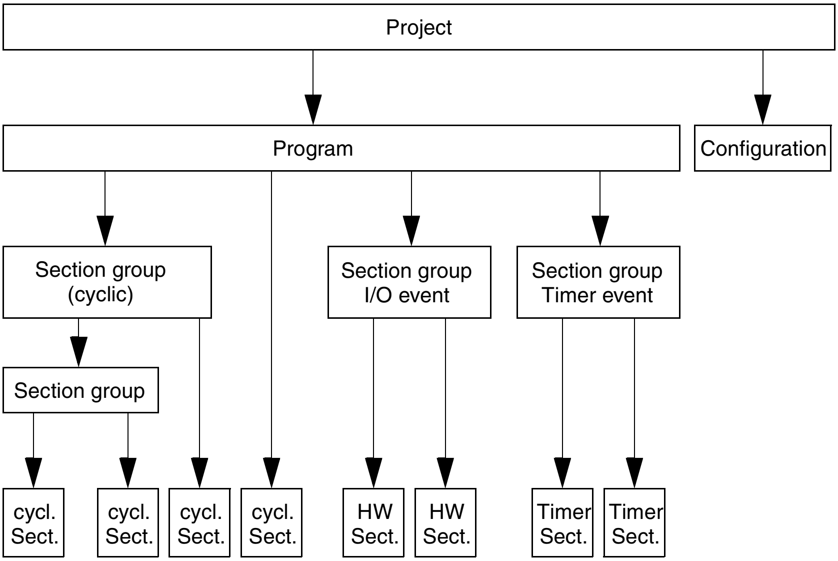
Project Structure and Processing

Structure of a project

The creation of a PLC program with Concept is carried out hierarchically in a project using PLC configuration (see *Configuration data, p. 53*) and Program (see *Programs, p. 42*). The program is divided into section groups and Sections (see *Sections, p. 47*).

The PLC configuration and required program parts can be created in any order within a project (top down or bottom up).

Structure of a project:



Processing an IEC/LL984 project

This table describes the processing of a LL984/IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	Executing LL984 segment 1	Writing outputs calculated in segment n
		Reading inputs required in segment 2
3	Executing LL984 segment 2	Writing outputs calculated in segment 1
		Reading inputs required in segment 3
4	Executing LL984 segment 3	Writing outputs calculated in segment 2
		Reading inputs required in segment 4
...	...	...
n	Executing LL984 segment n (n =< 32)	Writing outputs calculated in segment n-1
		Reading inputs required in segment 1
n+1	Executing IEC section 1	-
n+2	Executing IEC section 2	-
n+3	Executing IEC section 3	-
	..	-
m	Executing IEC section n (n =< 1600) and back to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - 4 In these stages, the logic for the LL984 sections is executed by the logic processor in segments 1-3 (corresponding to the settings in the Segment scheduler (see *Segment manager, p. 99*)).
At the same time the I/O processor transfers the output values calculated in the respective previous segment to the hardware and the hardware reads the input values required for the next respective segment.

n In this step, the logic processor in segment n runs the LL984 sections logic.
At the same time the I/O processor transfers the output values calculated in the previous segment to the hardware and the hardware reads the input values required for segment 1.
Note: The output values calculated in this segment are only executed on next execution of stage 2, i.e. after the IEC logic and the overhead have been processed. Therefore no time critical logic should be executed in this segment.

n+1 - m The logic processor runs the IEC sections logic in these steps.
It then "jumps back" to stage 1.
Note: No hardware signals are read or written. The values calculated/read in stages 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the segment scheduler).

Processing a LL984 project

This table describes the processing of a LL984 project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	Executing LL984 segment 1	Writing outputs calculated in segment n
		Reading inputs required in segment 2
3	Executing LL984 segment 2	Writing outputs calculated in segment 1
		Reading inputs required in segment 3
4	Executing LL984 segment 3	Writing outputs calculated in segment 2
		Reading inputs required in segment 4
...	...	...
n	Executing LL984 segment n (n =< 32) and back to stage 1	Writing outputs calculated in segment n-1
		Reading inputs required in segment 1

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - 4 In these stages, the logic for the LL984 sections is executed by the logic processor in segments 1-3 (corresponding to the settings in the Segment scheduler (see *Segment manager, p. 99*)).
At the same time the I/O processor transfers the output values calculated in the respective previous segment to the hardware and the hardware reads the input values required for the next respective segment.

n In this step, the logic processor in segment n runs the LL984 sections logic.
At the same time the I/O processor transfers the output values calculated in the previous segment to the hardware and the hardware reads the input values required for segment 1.
It then "jumps back" to stage 1.
Note: The output values calculated in this segment are only processed the next time stage 2 is completed, i.e. after the overhead has been processed. Therefore no time critical logic should be executed in this segment.

Processing an IEC project

This table describes the processing of an IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	-	Writing outputs allocated to segment 1
		Reading inputs allocated to segment 1
3	-	Writing outputs allocated to segment 2
		Reading inputs allocated to segment 2
4	-	Writing outputs allocated to segment 3
		Reading inputs allocated to segment 3
...	...	...
n	-	Writing outputs allocated to segment n (n =< 32)
		Reading inputs allocated to segment n (n =< 32)
n+1	Executing IEC section 1	-
n+2	Executing IEC section 2	-
n+3	Executing IEC section 3	-
	..	-
m	Executing IEC section n (n =< 1600) and back to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - n The hardware signals from the allocated modules respective segments are written and read by the I/O processor in these stages (corresponding to the settings in the Segment scheduler (see *Segment manager, p. 99*)).

n+1 - m The logic processor runs the IEC sections logic in these steps.
It then "Returns" to stage 1.
Note: No hardware signals are read or written. The values read in stage 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the Segment manager).

Processing an IEC project

This table describes the processing of an IEC project (Quantum):

Step	Logic processor	I/O processor
1	Overhead, e.g. communication with NOM, NOE etc.	-
2	-	Writing outputs allocated to segment 1
		Reading inputs allocated to segment 1
3	-	Writing outputs allocated to segment 2
		Reading inputs allocated to segment 2
4	-	Writing outputs allocated to segment 3
		Reading inputs allocated to segment 3
HE1	1. I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
HE2	2. I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
...	...	...
HE64	64. (last) I/O event section, spontaneous execution, when Hardware Interrupt occurs	-
TE1	1. Timer event section, only executed when time interrupt occurs	-
TE2	2. Timer event section, only executed when time interrupt occurs	-
...	...	...
TE16	16. Timer event section, only executed when time interrupt occurs	-
...	...	...
n	-	Writing outputs allocated to segment n (n =< 32)
		Reading inputs allocated to segment n (n =< 32)
n+1	Executing IEC section 1 (cyclically)	-
n+2	Executing IEC section 2 (cyclically)	-
n+3	Executing IEC section 3 (cyclically)	-
	..	-
m	Executing IEC section n (n =< 1600) and return to stage 1	-

1 The overhead is executed in this stage (e.g. communication with the coupling modules NOM, NOE).

2 - n The hardware signals from the allocated modules respective segments are written and read by the I/O processor in these stages (corresponding to the settings in the Segment scheduler (see *Segment manager, p. 99*)).

n+1 - m The logic processor processes the IEC sections logic in these steps.
It then "Returns" to stage 1.
Note: No hardware signals are read or written. The values read in stage 2 to n are used exclusively. The outputs calculated in these stages are transferred in stages 2 to n (corresponding to the settings in the Segment scheduler).

HE1 - HE64 If a hardware interrupt signal specially assigned to a section changes its value according to its parameter configuration, the cyclical processing and if necessary the processing of a Timer event section is immediately stopped and returned to the I/O event section. Once all event sections (and Timer event sections) are processed, the cyclical processing is continued at the point where the interrupt occurred. (See also chapter "*I/O Event Sections, p. 1154*")

TE1 - TE16 When a specially configured Timer interrupt signal for a section occurs, cyclical processing is immediately stopped and jumps to the Timer event section. Once Timer event sections are processed, the cyclical processing is continued at the point where the interrupt occurred as long as there are no further instructions for Timer event sections. (See also chapter "*Timer Event Sections, p. 1140*")

Programs

Structure of a program

A program consists of one or more Sections (see *Sections, p. 47*) or section groups. Section groups can contain sections and other section groups. Section groups can be created exclusively and filled using **Project** → **Project browser (see *Project Browser, p. 549*)**. Sections describe the entire systems mode of operating.

Moreover the variables, constants, literals and direct addresses are managed within the program.

Variables

Variables are used to exchange data within a section, between several sections and between the program and the PLC.

Variables are declared using the menu command **Project** → **Variable declaration**. If the variable with this function is assigned an address, it is called a Located variable. If the variable has no address assigned to it, it is called an Unlocated variable. If the variable is assigned with a derived data type, it is called a Multi-element variable.

There are also constants and literals.

The following table provides an overview of the various types of variables:

Variable type	**Description**
Located variables	Located variables are allocated a State RAM address (reference address 0x, 1x, 3x,4x). The value of this variable is saved in the State RAM and can be changed online using the Reference data editor. These variables can be addressed using their symbolic names or using their reference address. All PLC inputs and outputs are connected to the State RAM. The program can only access peripheral signals attached to the PLC via located variables. Access from external pages via Modbus or Modbus Plus interfaces of the PLC, e.g. from visualization systems can be made using located variables.
Unlocated variables	Unlocated variables are not assigned a State RAM addresses. They therefore do not occupy any State RAM addresses. The value of this variable is saved internally in the system and can be changed using the Reference data editor. These variables are only addressed using their symbolic names. Signals requiring no peripheral access, e.g. intermediate results, system tags etc, should primarily be declared as unlocated variables.
Multi element variables	A variable which is assigned a Derived data type. A distinction is made here between Structured variables and Array variables.
Structured variables	Variables to which a Derived data type defined using a STRUCT (structure) is assigned. A structure is a collection of data elements with generally different data types (Elementary data types and/or Derived data types).
Array variables	A variable which is assigned a defined data type with the key word ARRAY. An array is a collection of data elements with the same data type.

Variable start behavior

In start behavior of PLCs there is a distinction between cold restarts and warm restarts:

- **Cold restart**
  Following a cold restart (loading the program with **Online** → **Download**) all variables (irrespective of type) are set to "0" or their initial value if available.
- **Warm restart**
  In a warm restart (stopping and starting the program or **Online** → **Download changes**) different start behaviors are valid for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    In a warm restart all 0x, 1x and 3x registers are set to "0" or their initial value if available.
    The buffered coils are an exception to this. Buffered coils retain their current value (storage behavior).
    4x registers retain their current value (storage behavior).
  - **Unlocated variables**
    In a warm restart all unlocated variables retain their current value (storing behavior).

This varying behavior in a warm restart leads to peculiarities in the warm restart behavior of set and reset functions.

- **Set and Reset in LD and IL**
  Warm restart behavior is dependent on the variable type used (storage behavior in use of unlocated variables; non storage behavior in use of located variables/direct addresses)
- **SR and RS Function Blocks in FBD, LD, IL and ST**
  These function blocks work with internal unlocated variables and therefore always have a storage behavior.

Constant variables

Constants are unlocated variables assigned a value, which cannot be modified by the logic program (read only).

Literals (values)

Literals are used to describe FFB inputs, and transition conditions etc using direct values. These values cannot be overwritten by the program logic (read only).

The values of literals can be changed online.

There are two different types of literal; generic and standardized.

The following table provides an overview of the various types of literals:

Literal	Description
Generic literals	If the literal's data type is not relevant, simply specify the value for the literal. In this case, Concept automatically assigns a suitable data type to the literal.
Standardized literals	If you would like to manually determine a literal's data type, this may be done using the following construction: "Data type name"#"Literal value" For example INT#15 (Data type: Integer, value: 15), BYTE#00001111 (Data type: Byte, value: 00001111) REAL#23.0 (Data type: Real, value: 23.0) To assign the data type **REAL** the value may also be specified in the following manner: 23.0. Entering a comma will automatically assign the data type REAL.

Direct addresses

Direct addresses are memory ranges in the PLC. They are located in the State RAM and can be assigned Input/Output modules.

Direct addresses can be entered or displayed in various formats. The display format is specified in the dialog box **Options** → **Preferences** → **Common**. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:

- **Standard format (400001)**
  The five character address comes directly after the first digit (the Reference).
- **Separator format (4:00001)**
  The first digit (the Reference) is separated from the following five-character address by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the following address by a colon (:), and the leading zeros of the address are not given.
- **IEC format (QW1)**
  In first place, there is an IEC identifier, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

The values of direct address can be modified online using the Reference data editor (see *Reference data editor, p. 587*).

Start behavior of digital outputs

Outputs that are assigned 0x registers are deleted during PLC startup. Digital outputs that assigned 4x registers keep their current value when the PLC is stopped or started.

Sections

Introduction

A program consists of one or more sections. A section describes the mode of functioning of a systems technological unit (for example a motor).

Each section has its own document window in Concept. For overview purposes it is useful to divide a very large section into several small ones. The scroll bar is used to move within a section.

The page break can be made visible for each section, so that the page format can be monitored when programming. In this way, a readable printout of the section is assured.

Section types

There are three different types of sections in Concept provided for Quantum processing.

- **Cyclical section** are executed in every program cycle. The reaction time depends on the cycle time and is a minimum of one cycle and maximum of two cycles.
- **I/O event sections** are not executed cyclically, but are started and processed spontaneously when a specially assigned Interrupt signal value changes state (corresponding to the setting in the Configurator and Section properties).
  The 140-HLI-340-00 module provides 16 Interrupt inputs. The local backplane has space for a maximum of 4 HLI modules.
  The reaction time to an I/O event generally depends on the process duration of the EFBs to be processed in the section as well as the transition times.
- **Timer event sections** are started and processed in precise user defined intervals.
  The time intervals are defined in multiples of 1ms and a Phase in the Section properties for Timer Event Sections dialog box.
  The reaction time is independent of the cycle time. Reactions to outputs are also carried out in defined time intervals.

Maximum number of sections

There can be up to a maximum of 1,600 sections per program.

Programming languages

Sections can be programmed using the IEC programming languages FBD (Function Block Diagram), LD (Ladder Diagram), SFC (Sequential Control), IL (Instruction List), or ST (Structured Text), or in the LL984 programming language (Ladder Logic), which resembles Modsoft. Only one of the stated programming languages is permitted to be used within a section.

Exchanging values

Values are exchanged within sections via links, variables, or direct addresses. Values are exchanged between different sections via variables or direct addresses.

Section execution order

The LL984 sections are the first to be executed. The LL984 section vertical sequence can be defined via the **Project** → **Configurator** → **Configure** → **Segment scheduler...** dialog box. Once the entire LL984 section has been processed, the IEC sections are then processed (FBD, SFC, LD, IL, ST). The execution order can be determined using either the **Project** → **Execution order...** or the Project browser (see *Project Browser, p. 549*) dialog box.

Printing sections

Sections are divided into pages when printing out. The amount of information on these pages is dependent on the settings in the menu **File** → **Print**. Page division can be displayed using the menu option **View** → **Page breaks**.

Section variable

A Multi-element variable is automatically generated for each IEC section (FBD, SFC, LD, IL, and ST) and has the same name as the section.

This variable is SECT_CTRL data and has two elements:

- The "disable" BOOL data type element for disabling sections.
- The "hsbyState" BYTE data type element for displaying the Hot Standby status of sections.
  If the smallest bit of this element is set, the data from this section is transferred/received, see the *Hot Standby User's manual*. (This bit corresponds to the exclamation mark in the project browser.)

Disabling sections

The component "disable" can be used to enable/disable the section variable If the multi element address is not used or if the value 0 has been assigned to "disable", the corresponding section is executed. If "disable" is assigned the value "1", the corresponding section will not be executed. By using this variable, the execution of sections can be controlled according to events.

Note: If a disabled section is animated, the DISABLED status is displayed in the status bar.

⚠ CAUTION

Risk of unwanted process states.

Disabling a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section is disabled. The status of these outputs cannot be modified.

Failure to follow these instructions can result in injury or equipment damage.

Disabling Interrupt Sections

A specific Interrupt section can be disabled using the ISECT_OFF block. It can be enabled again using the ISECT_ON block. The section names are provided by the SECT_CTRL control variable.

The I_LOCK block can disable all interrupt sections. They can be enabled again using the I_UNLOCK block.

Note: A possible interrupt on an interrupt section has no effect.

Lock section UNCONDITIONALLY (possibility 1)

The procedure for locking a section unconditionally is as follows:

Step	Action
1	Using **Online** → **Reference data editor** open the Reference data editor (see *Reference data editor, p. 587*).
2	By double clicking on a line number, open the **Lookup variables** dialog box.
3	From the area **Data type** first choose the option **Structured** and then from this list **SECT_CTRL**. **Result:** The names of all sections are displayed.
4	Now select the names of the section to be locked.
5	Use the command button **Components...** to select the **ANY type components** dialog box.
6	Select the line **disable: BOOL** and confirm with **OK**.
7	If the following has not been performed yet: Create a connection between the PLC and the programming device and load your program onto the PLC.
8	Change the entry in the column **Value** to 1 (TRUE) to lock the section or 0 (FALSE) to enable the section.
9	Using **Online** → **Animation** activate the animation if it is inactive. **Result:** The section is disabled or enabled according to the value. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

⚠ CAUTION

Risk of unwanted process states.

The entry in the column **Value** remains even after the reference data editor has been closed (even if the entries are not saved), or in other words, the section remains disabled and must be explicitly re-enabled via the reference data editor (value = 0).

Failure to follow these instructions can result in injury or equipment damage.

Lock section UNCONDITIONALLY (possibility 2)

The procedure for locking a section unconditionally is as follows:

Step	Action
1	Using **Project** → **Project browser** open the Project browser (see *Project Browser, p. 549*).
2	From **Online** → **Connect...** create a connection between the programming device and the PLC.
3	From **Online** → **Download...** (if the program is in **NOT EQUAL**mode) or **Online** → **Download changes** (if in **MODIFIED** mode) restore the consistency between the programming device and the PLC.
4	Select the section to be locked from the project browser.
5	Activate the context menu for sections using the right mouse button, and activate **Animate enable state**.
6	Change the enable status using the menu command **Switch enable state** from the context menu (right mouse button) of the selected section. **Note:** Sections may only be disabled or enabled via the Project browser, if they have not already been disabled/enabled via another Section (see *Locking a section CONDITIONALLY, p. 52*) or via the Reference data editor (see *Lock section UNCONDITIONALLY (possibility 1), p. 50*). **Result:** The section is locked. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

Locking a section CONDITIONALLY

The procedure for locking a section conditionally (program dependent) is as follows:

Step	Action
1	Create the logic according to the section to be locked, for example in an FBD section. When doing this, please note that the logic must carry a BOOL data "output" and that the section to be disabled will be disabled at logic "1". **Note:** The section containing a logic for disabling/enabling other sections should not be disabled.
2	By double clicking on your logic's "output", open the **Connect FFB** dialog box.
3	Use the command button **Lookup...** to open the **Lookup Variable** dialog box.
4	From the area **Data type** first choose the option **Structured** and then from this list **SECT_CTRL**. **Reaction:** The names of all sections are displayed.
5	By double clicking, now select the names of the section to be locked.
6	Select the line **disable: BOOL** and confirm with **OK**. **Result:** The multi-element variable from the section to be locked (Section name.disable) now creates the "output" of the logic.
7	From **Project** → **Execution order...** open the **Section Execution Order** dialog box.
8	Using the command buttons, ensure that the section containing the logic for locking is executed before the section to be locking is executed.
9	If the following has not been performed yet: Create a connection between the PLC and the programming device.
10	Download your program to the PLC. **Result:** When logic "1" is at the "Output" the section to be locked is not edited. **Note:** Locking a section does not mean that programmed outputs will be deactivated within the section if an output has already been set in a prior cycle, this status remains even after the section has been disabled. The status of these outputs cannot be modified.

Configuration data

Description

The PLC configuration is the interface between the program and the hardware.

The configuration data consists essentially of the component list and the entry in the address field of the program.

Loadables facilitate communication with the IEC programming language and the loading of further LL984-Instructions.

Creating a Project

4

At a Glance

Overview

This chapter describes the general procedure for the initial creation of a project. The most linear sequence possible is used here, in order to show a Concept-newcomer an easily manageable way of creating a project. Crosslinks between the Menu Commands are of course possible. As they gain experience, users will learn shortcuts and alternatives. For more detailed information, please see the relevant chapters in the user manual.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Overview	56
Step 1: Launching Concept	57
Step 2: Configuring the PLC	58
Step 2.1: Required Configuration	59
Step 2.2: Optional Configuration	61
Step 3: Creating the User Program	64
Step 4: Save	66
Step 5: Perform Memory Prediction	67
Step 6: Loading and Testing	68
Step 7: Optimize and Separate	73
Step 8: Documentation	75

Overview

Project Creation

The creation of a project has 8 main steps:

Step	Action
1	**Launching Concept** (see *Step 1: Launching Concept, p. 57*) Launch Concept and start a new project.
2	**Configuring the PLC** (see *Step 2: Configuring the PLC, p. 58*) Set the hardware configuration.
3	**Creating the user program** (see *Step 3: Creating the User Program, p. 64*) Create new sections and create your program.
4	**Save** (see *Step 4: Save, p. 66*) Save your project
5	**Perform Memory Prediction** (see *Step 5: Perform Memory Prediction, p. 67*) Check the PLC memory workload.
6	**Loading and testing the project** (see *Step 6: Loading and Testing, p. 68*) Create a link between the PC and the PLC. Load the project in the PLC and start it. Test the program with the Online Test Function. Now eliminate any mistakes in the program! Load the altered sections into the PLC.
7	**Optimize and Separate** (see *Step 7: Optimize and Separate, p. 73*) It is now advisable to optimize the program storage capacity and to reload the optimized program into the PLC. After successfully loading, testing and (if necessary) optimizing, you may disconnect the PC from the PLC. The program will now run offline.
8	**Documenting** (see *Step 8: Documentation, p. 75*) Create a complete set of documentation of your project.

Notes

Note: The steps "Configuring the PLC" and "Creating the User Program" can be performed in either order. This means that the PLC configuration can also be changed after the creation of the program.

Note: In order to prevent loss of data, you should save your program regularly.

Step 1: Launching Concept

Launching Concept

The procedure for launching Concept is as follows:

Step	Action
1	Double click on the Concept icon to launch Concept.
2	Select **File** → **New Project**.
3	You can specify a new project path or accept the standard project path with the project name `namenlos.prj`. **Result:**The new project is opened. **Note:** If you select the standard project path with the project name `namenlos.prj` , you can save this project with a name at a later time *Step 4: Save, p. 66*. A saved project can be invoked with the **Open Project...**, or by using its project icon.

Note

Note: For additional steps please note the settings in the submenu **Options** → **Preferences**!

Resume

Now proceed with Step 2: Configuring the PLC (see *Step 2: Configuring the PLC, p. 58*).

Step 2: Configuring the PLC

What should be configured?

Using **Project** → **PLC configuration** configure the entire hardware configuration for your project.

Required Configuration

Note: The PLC type must first be set! All further configurations can then be executed independently of the processing sequence.

The following configurations are necessary for the configuration:

- *Specifying the type of PLC (minimum configuration), p. 59*
- *Set memory partitions, p. 59*
- *Install loadables, p. 60*
- *Set I/O map, p. 60*

Optional Configuration

The following configurations are to be used according to the project:

- *Set head setup, p. 61*
- *Set Modbus communication , p. 61*
- *Set Peer Cop communication , p. 62*
- *Set data protection, p. 62*
- *Various PLC settings, p. 63*
- *ASCII messages (only for 984 LL), p. 63*

Step 2.1: Required Configuration

Precondition

The PLC type must first be set! All further configurations can then be executed independently of the processing sequence.

Specifying the type of PLC (minimum configuration)

The procedure for specifying the type of PLC (minimum configuration) is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **PLC Selection** menu command from the list. **Response:** The **PLC selection** dialog is opened.
3	From the **PLC family** list select your PLC type.
4	Select your CPU from the **CPU/Executive** list.
5	From the **Runtime** list select the **Enable** status. **Response:** It is possible to program sections in IEC languages (FBD, LD, IL and ST). **Note:** In the **Runtime** list, the status **Not available**, **Disabled** or **Only 984** is displayed, then the selected CPU does not support any IEC programming languages. If in the list the status **Only IEC** is displayed, then the selected CPU exclusively supports IEC languages and these do not have to be explicitly enabled.
6	With simple tests and programs the configuration can now be exited and the procedure continued from *Step 3: Creating the User Program, p. 64* or*Step 4: Save, p. 66*.

Set memory partitions

The procedure for setting the memory partition is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **PLC memory partition** menu command from the list. **Response:** The **PLC memory partition** dialog is opened.
3	In the **Discretes** and **Words** ranges select the probable number of I/O flag bits and I/O words, to be required by the user program **Note:** The maximum address range, that must not be exceeded, can be read on the right-hand side of the dialog.

Install loadables

The procedure for installing the loadables is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Loadables** menu command from the list box. **Response:** The **Loadables** dialog is opened.
3	Select the loadable in the **Available:** list. **Note:** Loadables are assigned in the *Loadables, p. 96*section.
4	Select the **Install =>** command button. **Response:** The selected loadable is moved to the **Installed:** field.
5	Repeat the steps 3 and 4 until all the loadables required have been installed.

Set I/O map

The procedure for setting the I/O map is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **I/O map** menu command from the list. **Response:** The **I/O map** dialog is opened.
3	Select the **Supervision time** column and enter a time, within which a communication exchange must take place. If this time is exceeded, an error message appears.
4	Select the **Edit...** command button. **Response:** The dialog for entering modules is opened.
5	In the **Module** column, select the **...** command button. **Response:** The **I/O Module Selection** dialog is opened.
6	In the **Modules** column, select the module. **Response:** The module is displayed in the current slot.
7	Select the **Input start** and/or **Output start** columns and enter the first address of the occupied input and/or output reference range for the module.
8	Select the module and choose the **Params**command button. **Response:** If the module has a parameter dialog, you can define the parameter (e.g. disconnect behavior, data format, measuring range) here.

Resume

Now proceed with Step 3: Creating the user program (see *Step 3: Creating the User Program, p. 64*).

Step 2.2: Optional Configuration

General Information

The following configurations do not need to be executed urgently, but they offer extended functions.

Set head setup

The procedure for specifying the remote I/O is as follows (this procedure is optional for minimum configuration):

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **I/O map** menu command from the list. **Response:** The **I/O map** dialog is opened.
3	Select the **Head setup...** command button. **Response:** The **Head Setup** dialog is opened.
4	Enter the slots for the RIO or NOM modules. **Response:** Return to the **I/O map** dialog.
5	Select the head setup in the **Go To** list.
6	Select an empty line (last line) in the table, and select the **Insert**command button. **Response:** In the **Type** column another I/O station is entered.
7	Select the **Drop** column and enter the station number. **Note:** Only as many remote I/O stations can be configured as there are segments registered in the segment scheduler.
8	Select the head setup in the **Go To** list for the 2nd drop.
9	Next, carry out steps 3 to 6 of the *Set I/O map, p. 60* procedure.

Set Modbus communication

To set the Modbus communication (Quantum slave, terminal, printer, etc.) proceed as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Modbus Port settings** menu command from the list. **Response:** The **Modbus port settings** dialog is opened.
3	Make the corresponding settings.

Set Peer Cop communication

If a Modbus Plus link exists, the Peer Cop functionality is able to transfer state RAM data globally or directly between several nodes on a local network. The procedure for setting the Peer Cop communication is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Config. Extensions** → **Select Extensions** list. **Response:** The **Select extensions** dialog is opened.
3	Check the **Peer Cop** box. **Response:** Return to the **PLC configuration** window and the **Peer Cop** menu command is now available.
4	Select **Config. Extensions** → **Peer Cop**. **Response:** The **Peer Cop** dialog is opened.
5	In the **Go To** range select the local bus devices, and enter the slot.
6	Select in the **Global** range the **Receive...** and **Send...** command buttons to define the destination and source addresses of the transmission data and/or the address of the other bus devices.
7	Select in the **Specific** range the **Receive...** and **Send...** command buttons to define the destination and source addresses of the transmission data and/or the address of the other bus devices.

Set data protection

Address ranges of coils and holding registers can be protected from being overwritten by external signals. The procedure for setting the data protection is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Config. Extensions** → **Configuration extensions**. **Response:** The **Configuration extensions** dialog is opened.
3	Check the **Data protection** box. **Response:** Return to the **PLC configuration** window and the **Data protection** menu command is now available.
4	Select **Config. Extensions** → **Data protection**. **Response:** The **Data protection** dialog is opened.
5	Select the range for the coils and holding registers. This range should contain write-protection.

Various PLC settings

Diverse internal PLC data can be evaluated, a watchdog timeout for the user program can be specified, the time windows for the communication (I/O time disk) parameterized and the multiple assignment of outputs authorized. The procedure for setting the PLC settings is as follows:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select the **Specials** menu command from the list. **Response:** The **Specials** dialog is opened.
3	Check the **Battery coil**, **Timer register** and **Time of Day** check boxes and enter an address in the corresponding text boxes.
4	Check the **Allow Duplicate Coils** check box and enter the address from which this should be allowed in the text box..
5	In the **Watchdog timeout (ms*10):** text box enter a numeric value between 2 and 255 (ms). This enables you to set an impulse watchdog for the user program. **Response:** As soon as the count pulses exceed the specified time, an error message appears.
6	In the **Online Editing Timeslice (ms):** text box enter a numeric value between 3 and 100 (ms). This enables you to define a time for executing the multi-cycle edit functions (paste, delete, find etc.)

ASCII messages (only for 984 LL)

To set the ASCII messages (only for 984LL), execute the following steps:

Step	Action
1	Select **Project** → **PLC configuration**. **Response:** The **PLC configuration** window is opened, this contains further menu commands for hardware configuration.
2	Select from the list **ASCII** → **ASCII Setup**. **Response:** The **ASCII Setup** dialog is opened.
3	Enter the total messages, the size of the message width and the number of ASCII ports (from the I/O periphery) in the text boxes. **Response:** In the **PLC configuration** → **ASCII** window the **ASCII Port Settings** menu command is available.
4	Select from the list **ASCII** → **ASCII port settings**. **Response:** The **ASCII port settings** dialog is opened.
5	Make the corresponding settings. **Note:** ASCII messages can now be created under **Project** → **ASCII messages...** .

Resume

Now proceed with Step 3: Creating the user program (see *Step 3: Creating the User Program, p. 64*).

Step 3: Creating the User Program

General

A user program is created in sections. Each section is programmable in one of the available languages and has a unique name in the project. Sections can be generated at any time during the programming.

Overview

The creation of a user program consists of 9 steps:

Step	Action
1	Generating a New Section (see *Generating a New Section, p. 64*)
2	Declaring the Variables (see *Declaring the Variables, p. 64*)
3	Programming a Section (see *Programming a Section, p. 65*)
4	Analyzing Program/Section (see *Analyzing Program/Section, p. 65*)
5	Specifying the section execution sequence (see *Set execution order of sections, p. 65*)

Generating a New Section

The procedure for generating a new section is as follows:

Step	Action
1	In the main menu **File** call up the menu command **New section...** . **Result:** The dialog box **New program section** is opened.
2	Click on the programming language desired for this section.
3	In the text box **Section name** enter the unique name for this section.
4	Generate all the required sections in this way.

Declaring the Variables

A program consists of functions and Function Blocks (FFBs) or of instructions with the statement of variables (e.g. signals), addresses or literals. While direct addresses and literals can be used immediately, variables must be declared before they can be used in programming. The procedure for declaring variables is as follows:

Step	Action
1	In the main menu **Project** call the menu command **Variable declaration...** . **Result:** The dialog box **Variable declaration** is opened.
2	Enter the variable name, the associated data type, and if necessary the reference address, the initial value and a comment.
3	Confirm the entries with **OK**. **Note:** Further editing is also possible from a FFB connection or contact etc. by double-clicking -> **Var. Declaration...** . This starts the Variables editor.

Programming a Section

The procedure for programming a section is as follows:

Step	Action
1	Using **File** → **Open section** open the section to be programmed.
2	Create programs according to the rules of the individual programming languages: ● Function Block Diagram FBD (see *Function Block language FBD, p. 195*) ● Ladder Diagram LD (IEC) (see *Ladder Diagram LD, p. 223*) ● SFC (Sequential Control) (see *Sequence language SFC, p. 257*) ● Instruction list (IL) (see *Instruction list IL, p. 307*) ● Structured text (ST) (see *Structured text ST, p. 377*) ● LL984 (Ladder Diagram (Modsoft)) (see *Ladder Logic 984, p. 439*)

Analyzing Program/Section

Check a section or the entire program for syntax violations! The procedure for analyzing a program/section is as follows:

Step	Action
1	In the main menu **Project** call up the menu command **Analyze section** or **Analyze program**.
2	Remove the cause of the displayed or reported error. **Note:** Loading a section or program into the PLC is only possible after an error-free check. (The removal of the cause of warnings is not absolutely necessary. Checking the warnings is, however, sensible.)

Set execution order of sections

The sections are initially stored in the order of their creation and are executed after the program has started. In general this sequence must be adjusted project-specifically to suit the task setting. The procedure for specifying the section execution sequence is as follows:

Step	Action
1	To specify the section execution sequence there are two alternatives: ● In the main menu **Project** call the menu command **Execution order...** and using the command buttons **First**, **Last**, **Next**, **Previous** sequence the sections as required. ● In the main menu **Project** call up the menu command **Project browser** and sequence them as required by moving them around in the *Project Browser, p. 549*.

Resume

Now proceed with Step 4: Saving (see *Step 4: Save, p. 66*).

Step 4: Save

General Information

General information about saving:

- If you exit a project without saving, you will be automatically asked if you want to save the project or not. If you answer yes to this question, this begins the same procedure described below.
- In order to prevent loss of data, projects should be saved regularly during long periods of configuration or programming sessions.

Saving a Project for the First Time

The procedure for saving a project for the first time is as follows:

Step	Action
1	In the **File** main menu invoke the **Save Project As...** menu command.
2	In the **File name** text box, enter the project name name.prj.
3	Select the desired drive and directory from the **Directory** list. Alternatively, it is possible to enter the whole path specification in the **File name** text box, e.g. **`c:\product1\reactor3.prj`** `(max. 28 characters + .prj).` If these directories do not yet exist, they will be automatically created. **Note:** According to IEC 1131, a project includes all programs, data etc which belong to a PLC. If several projects (i.e. PLCs) belong to one system, then all projects should be stored in a common directory named after the system.
4	Click the **OK** command button. **Response:** The project has now been stored in the specified directory under the given name.

Supplementary Saving

The procedure for supplementary saving is as follows:

Step	Action
1	From the **File** main menu simply select the **Save** menu command.

Resume

Now proceed with Step 5: Executing memory prediction (see *Step 5: Perform Memory Prediction, p. 67*).

Step 5: Perform Memory Prediction

Check the PLC memory workload.

Perform an offline memory prediction of the configured PLC before downloading the program to the PLC. The table displayed in the **Project** → **Memory Prediction** dialog shows the use of individual memory ranges. An expected memory workload is then recognized.

> **Note:** In some cases the memory prediction is not very accurate. A discrepancy between required memory in the PLC and the memory prediction under Concept may occur. The memory prediction always indicates more available memory than is actually available in the PLC.
> This is due to the dynamic memory in the DFBs and Sections, which is difficult to calculate. Especially ST sections cause a great difference between the prediction and PLC. To be sure that there is sufficient memory available in the PLC, load a project into a PLC for examination. The simulator cannot be used because many projects have sufficient memory in the simulator but not in the PLC.

Resume

Now proceed with Step 6: Loading and testing the project (see *Step 6: Loading and Testing, p. 68*).

Step 6: Loading and Testing

General Information

Loading and testing programs is only possible if

- either the 16-bit simulator Concept SIM is switched on or
- the Concept SIM 16-bit simulator is switched off and a PLC is attached with a Modbus Plus, Modbus, TCP/IP cable, or
- the Concept PLCSIM32 simulator is switched on.

Note: Testing using Concept SIM (see *Simulating a PLC (16-bit simulator), p. 753*) and Concept PLCSIM32 (see *Simulating a PLC (32-bit simulator), p. 755*) simulators is only possible with IEC user programs.

Overview

Loading and testing macros is divided into 9 main steps:

Step	Action
1	Loading the EXEC file into the PLC (see *Concept Installation Instructions*)
2	Connecting the PC and PLC (see *Connecting the PC and PLC, p. 68*)
3	Loading and Starting the Program (see *Loading and Starting the Program, p. 69*)
4	Activating the Animation (see *Activating the Animation, p. 70*)
5	Changing the Values of Literals (see *Changing the Values of Literals, p. 70*)
6	Changing the Values of Variables (see *Changing the Values of Variables, p. 71*)
7	Locating Errors (see *Locating Errors, p. 71*)
8	Downloading Changes (see *Downloading Changes, p. 72*)
9	Starting and Stopping the PLC (see *Starting and Stopping the PLC, p. 72*)

Connecting the PC and PLC

The procedure for linking the PC and the PLC is as follows:

Step	Action
1	From the **Online** main menu invoke the **Connect...** menu command. **Response:** The **Link to PLC** dialog box opens.
2	Set the protocol type (Modbus, Modbus Plus, TCP/IP or Simulator) and the PLC node (when working in a network) with which you wish to communicate.
3	Under **Access right** select the **Change Configuration** option
4	Confirm the details with **OK**.

Loading and Starting the Program

The procedure for loading and launching the program is as follows:

Step	Action
1	From the **Online** main menu invoke the **Connect...** menu command. **Response:** The **Download Controller** dialog box will be opened in the PLC.
2	When loading the program for the first time, use the **All** command button.
3	Click the **Load** command button. **Response:** Various dialog boxes will be displayed.
4	Answer the question `Stop the program in PLC? Yes/No` with **Yes**. **Note:** This question only appears when a program is already running in the PLC.
5	Answer the question `Start a program in PLC? Yes/No` with **Yes**, if there are no errors. If warnings or errors are reported, these will be listed in the **Messages** window. Correct the warnings or errors at the specified point.

Activating the Animation

With the animation (online status report) it is possible to monitor the status of variables, steps, transitions etc within individual sections of the editor window. The procedure for activating the animation is as follows:

If…	Then…
To display binary values exclusively.	To display binary values exclusively, invoke the **Online** main menu and click on the **Animate booleans** menu command. **Response:** The valences of all booleans (variables, direct addresses, literals) are displayed in colour (0-Signal = red, 1-Signal = green).
If you want to display the values of all variables.	To display the values of all variables invoke the **Editing** main menu option and select the **Select All** menu command (selects all items in the current section). Thereafter invoke from the **Online** main menu option the **Animate selection** menu command. **Response:** The valences of all values (variables, direct addresses, literals) are displayed in colour (red = 0-Signal, green = 1-Signal, yellow = either, for variables, immediate display of the value or, for multi-element-variables, displays the value by double-clicking on the variable).
If you want to enter monitoring fields in the text languages (IL and ST).	Use the **Selected Inspect** menu command to paste the text languages IL and ST into section monitoring fields. **Response:** The current value of the allocated variables is shown in these monitoring fields. With multi element variables, only the value of the first element is shown. This can be changed by double-clicking on the monitoring field of the **Numeric Inspect Settings** dialog box, which invokes the options available.

Changing the Values of Literals

The procedure for changing literals is as follows:

Step	Action
1	Activate the animation, as described in *Activating the Animation, p. 70*.
2	Double-click on the literal to be changed.
3	Enter a new value and confirm with **OK**. **Response:** The new value will be sent to the PLC during the next logic scan.

Changing the Values of Variables

With the Reference data editor (see *Reference data editor, p. 587*) it is possible to show and set the values of variables (state, control, force). The procedure for changing variables is as follows:

Step	Action
1	From the main menu, select **Online** and then the **Reference data editor** menu command.
2	Enter the variables to be displayed in the dialog box marked **RDE Templates**.
3	To set the value highlight the **Disable** check box, and enter the desired value.
4	The RDE template can be saved under a unique name. To do this, invoke the **RDE** main menu option and select the **Save template as…** menu command. **Note:** Several RDE templates can be invoked at once. To do this, invoke the **RDE** main menu option and select the **Open template...** menu command.

Locating Errors

If errors occur during the processing of the program by the PLC, these will generally be reported on screen **Messages** and entered in an events list in log book form. The procedure for locating errors is as follows:

Step	Action
1	From the **Online** main menu invoke the **Event Viewer** menu command. **Response:** A window is opened, in which all errors are listed and described.
2	Select an error line and use the command button **Go to Error**. **Response:** This will go directly to the section in which the error occurred. The faulty object is highlighted.
3	Correct the program.
4	If your program now has the **UNEQUAL** status carry out the steps in Downloading and Starting the Program (see *Loading and Starting the Program, p. 69*) once again. If the program now has the **MODIFIED** status perform the steps in Downloading Changes (see *Downloading Changes, p. 72*) once again.

Downloading Changes

If the project has the **MODIFIED** status after it has been altered, these changes can be loaded online into the PLC without stopping the program currently running. The procedure for downloading changes is as follows:

Step	Action
1	From the **Online** main menu access the **Download Changes...** menu command.
2	Click on **OK**. **Response:** The changes will be downloaded to the controller.

Starting and Stopping the PLC

The procedure for starting and stopping the PLC is as follows:

Step	Action
1	If the same project is running on the PC and PLC (**EQUAL**), then the PLC can be started or stopped with **Online** → **Online Control Panel...** .

Resume

Now proceed with Step 7: Optimize and Separate (see *Step 7: Optimize and Separate, p. 73*).

Step 7: Optimize and Separate

Optimizing Projects

At the end of the installation and/or after several runs of**Download Changes...** it is useful to perform an optimization, so that any gaps in the program data memory management are filled. After optimization the project is **UNEQUAL** on the PC and PLC and the program must be loaded into the PLC with **Download...** (**Warning:** Program must be stopped and restarted!). The procedure for optimizing projects is as follows:

Step	Action
1	Save the project with **File** → **Save Project**.
2	In the **File** main menu invoke the **Close project** menu command and take note of the dialog boxes which then appear.
3	In the **File** main menu invoke the **Optimize Project...** menu command and select the project to be optimized. Take note of the dialog boxes which subsequently appear.
4	Check the size of the program data memory in the **Online** main menu with the **Memory Statistics...** menu command.
5	The sizes can then be altered with **PLC configuration**.
6	Save the project with **File** → **Save Project**.
7	Reload the optimized program into the PLC using **Online** → **Download...** . To do this the program currently running must be stopped.
8	Start the newly loaded program using **Online** → **Online Control Panel**.

Separating the PC and Controller

After successfully testing the program in the PLC (with a connected process) the PC can be separated from the controller. The procedure for separating the PC and the controller is as follows:

Step	Action
1	Please take note of the program status in the footnote! To maintain consistency **EQUAL** must be there. ● if it reads**MODIFIED**, modifications must be loaded first *Downloading Changes, p. 72*. ● If it reads**UNEQUAL** the program must be reloaded into the PLC *Loading and Starting the Program, p. 69*.
2	From the **Online** main menu access the **Disconnect...** menu command. Take note of the information in the displayed dialog box.
3	The project can be closed after separation. In the **File** main menu invoke the **Close project...** menu command. Take note of the information in the dialog box, if displayed.

Resume

Now proceed with Step 8: Documenting (see *Step 7: Optimize and Separate, p. 73*).

Step 8: Documentation

General information

Each project should be fully documented. Changes and additions should also be documented (partial documentation).

Among other things documentation includes:
- Comments on the project (**Project** → **Properties**),
- Comments on each separate section (**File** → **Section properties**),
- Comments on variables,
- Comments on the functions applied, function modules and DFBs (command button **Comment** in the property dialog of each module),
- Comments on steps and transitions (command button **Comment** in the property dialog of each element),
- Comments in the form of freely placed text elements in the graphic programming languages (**Object** → **Text**),
- Comments on each line of commands in the textual programming languages
- Comments on user-specific data types,
- Comments on derived function modules (DFBs).

Printing the documentation

The procedure for printing documentation is as follows:

Step	Action
1	In the main menu call up **File** menu command **Print...** .
2	In dialog box **Documentation contents** select **Page layout** whether each page should have a uniform header and footer as well as printing a front page. The appearance of header, footer and front page is stored in the available ASCII files.
3	In the area**Contents** and in dialog box **Documentation contents**, select what is to be printed.
4	If **Variable list** has been selected, call up **Options** in order to select the variables which are to be printed.
5	When **Sections** has been selected, ● call up **Select** and specify the sections that are to be printed and ● also call up **Options**. In area **Graphics enlargement factor** also specify the appropriate size of the logic which is to be printed.
6	Activate command button **OK**. **Reaction:** All entries are saved.
7	Make sure that the page set-up of the sections is as desired. In the main menu call up **View**follow this with the successive menu commands **Overview** and **Pabe Break**.
8	Change the order of for example the FFBs in such a way, that there are as few transitions between adjoining pages as possible.
9	In the main menu call up **File** the menu command**Print...**again and activate command button **Print**. The printout is made with defined settings and the dialog box is closed.

PLC configuration

5

At a Glance

Overview

This section describes the single process for the hardware configuration.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	General information about hardware configuration	79
5.2	Configuration in OFFLINE and ONLINE mode	83
5.3	Unconditional Configuration	87
5.4	Optional configuration	105
5.5	Backplane Expander Config	117
5.6	Configuration of various network systems	121
5.7	Quantum Security Settings in the Configurator	133

5.1 General information about hardware configuration

At a Glance

Overview

This section contains general information about hardware configuration.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	80
Proceed in the following way with the configuration	81

General information

At a Glance

The system configuration has far-reaching consequences as it influences the entire control work mode. It has to define all control-specific information as well as general information, allocate the necessary memory space and determine the input/output area. For the first configuration the user must enter several basic details for the PLC area, such as PLC type and memory. Only valid configurations are authorized.

A configuration always refers to a Project, i.e. the menu command **PLC configuration** is only available when a project has been opened.

The configuration is available offline or online.

Proceed in the following way with the configuration

Introduction

In this section you are given a general overview on how to proceed with the configuration.

Use Configuration Menu

There are menu commands that absolutely must be carried out and are available in the **PLC Configuration** window. Grayed out menu commands are currently unavailable and can be enabled for extending the hardware-configuration in the **Config. Extensions** directory with the menu command **Select Extensions**.

Read in Module Set-up

The PLC module set-up is entered manually and can be compared with the connected hardware in ONLINE mode. After it has been read in, the modules missing in Concept are shown in the I/O map, and can be re-edited.

The I/O addressing must then be done for each module.

When doing this, please ensure the permitted references are used:

Modules	References
Analog input modules	3x references
Analog output modules	4x references
Digital input modules	3x or 1x references
Digital output modules	4x or 0x references
Expert modules - input	3x or 1x references
Expert modules - output	4x or 0x references

Downloading the Hardware Configuration

The hardware configuration of a project is saved and can be downloaded to the simulation program Concept-SIM, Concept-SIM32 or an automation installation. By doing this, the EQUAL status is established between the host computer and the PLC.

Note: The Concept-SIM must be deactivated for transfer of the configuration to a real PLC.

5.2 Configuration in OFFLINE and ONLINE mode

At a Glance

Overview

This section contains information for configuration in OFFLINE and ONLINE mode.

What's in this Section?

This section contains the following topics:

Topic	Page
General information	84
Available Functions in OFFLINE and ONLINE Modes	85

General information

At a Glance

In OFFLINE mode no link is created between programming device and PLC, and the configuration can be performed. In ONLINE mode there is a link between programming device and PLC, so that only one conditional configuration can take place.

Available Functions in OFFLINE and ONLINE Modes

Introduction

This section contains an overview of the available functions in OFFLINE and/or ONLINE mode. The possibilities in the ONLINE mode are different in their use of the simulator and the real PLC.

Configuration in OFFLINE Mode

In OFFLINE mode all menu commands are available for the hardware configuration in the **PLC Configuration** window. The submenus in the **Config. Extensions** directory can be enabled in the **Select Extensions** dialog to extend the configuration.

If the PLC is in ONLINE mode, you can switch to OFFLINE mode using the menu command **Online** → **Disconnect...**. In the footer of the editor window, the status-bar indicator NOT CONNECTED appears.

Configuration in ONLINE Mode and in the Active Simulator

A configuration is not possible in ONLINE mode with an active simulator or a Modbus Plus connection, i.e. no entries can occur. The available dialogs can only be invoked and read.

You can switch to ONLINE mode using the menu command **Online** → **Connect...** and establishing a connection between the host computer and the PLC.

Configuration in ONLINE Mode and in the Real PLC

Using the connection to a real PLC a configuration in ONLINE mode is possible, as long as the **Change Configuration** access level is activated.

It is not possible to configure or reconfigure a PLC while the PLC is in RUN mode. If a program is already running in the PLC, it must be stopped before reconfiguration can be implemented. Stop the PLC with **Online** → **Online Control Panel** → **Stop PLC**. After editing, the changes are automatically transferred to the hardware when the PLC is started up.

Note: When you delete an Expert module in ONLINE mode in the I/O map, the allocated loadable is also automatically deleted. If you wish to place this module back in the I/O map at a later time, it will be necessary to download again.

You can switch to ONLINE mode using the menu command **Online** → **Connect...** and establishing a connection between the host computer and PLC.

Effects of ONLINE Changes

If the following conditions are satisfied, all animated windows are automatically closed if a change is made in the I/O map (e.g. deleting or adding to a module)

Conditions:

- ONLINE mode
- animated section(s)
- Status between PLC and host computer is EQUAL
- Controller stopped
- Access level **Change Configuration** is activated.

5.3 Unconditional Configuration

At a Glance

Overview

This section contains a description of the configuration to be performed unconditionally and an overview of the presettings in the configuration menu.

What's in this Section?

This section contains the following topics:

Topic	Page
Precondition	88
PLC selection	89
CPU Selection for the PLC Type	90
PLC memory mapping	95
Loadables	96
Segment manager	99
I/O Map	101

Precondition

Introduction

Only when the CPU has been selected in the **PLC Selection** dialog will all the other menu commands become available in the **PLC Configuration** window.

The following dialogs are a minimum selection and MUST be edited as part of the hardware configuration.

- PLC Selection
- PLC Memory Partition
- Loadables
- Segment Scheduler
- I/O Map

The preferences can be adopted as long as they are compatible with the hardware being used.

PLC selection

Introduction

Select the PLC family (Quantum, Compact, Momentum or Atrium) and the CPU, as well as the memory size, according to use. All the available CPUs are listed in the list box.

Determine logic zone

The logic zone for the desired programming language (IEC or LL984) can be expanded to the corresponding PLC type with the PLC family selection.

The assignment and installation of the loadables is determined according to the following settings:

Selection	Meaning
Enable	Installation of the IEC loadables. A desired memory area for the IEC zone can be set up. The assignment and installation of the loadable pairing to the selected CPU is performed automatically in the **Loadables** dialog.
Disable	No installation of the IEC loadables. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984.
984 only/IEC only	Some Momentum CPUs can only be programmed in the IEC zone or only in the LL984 zone.

Determine total IEC memory

By defining the total IEC memory size and the global data, you also automatically determine the IEC-program memory size. On the basis of this size, the available memory space for the LL984 user program can also be determined.

Note: With global data it is the memory space of the unlocated variables.

Note: Total IEC memory = IEC program memory + global data

CPU Selection for the PLC Type

Introduction

When installing hardware (Concept EXECLoader), you are required to load various EXEC data files (*.BIN). This determines the firmware for various PLC types. The available PLC types, which can be operated by loading the EXEC data files with the corresponding CPUs, are shown in the following tables.

Loading Firmware for Quantum PLC Types

The following table shows the current EXEC versions, which are located on the Service Release CD and supplied with Concept.

Quantum PLC type:

140 CPU	Q186Vxxx (IEC+LL984)	Q486Vxxx (IEC+LL984)	Q58Vxxxx (IEC+LL984)	Q5RVxxxx (IEC+LL984)	QIECVxxx (IEC only) *	IEC memory (kbyte)
113 02	X (LL984 only)	-	-	-	-	
113 02S	-	-	-	-	X	max. 150
113 02X	X (LL984 only)	-	-	-	-	
113 03	X	-	-	-	-	max. 136
113 03S	-	-	-	-	X	max. 379
113 03X	X	-	-	-	-	max. 136
213 04	X	-	-	-	-	max. 305
213 04S	-	-	-	-	X	max. 610
213 04X	X	-	-	-	-	max. 305
424 0x	-	X	-	-	-	max. 465
424 0xX	-	X	-	-	-	max. 465
434 12	-	-	X	-	-	max. 890
534 14	-	-	X	-	-	max. 2550
434 12A (Redesigned CPU)	-	-	-	X	-	max. 890
534 14A/B (Redesigned CPU)	-	-	-	X	-	max. 2550

Note: * After the QIECVxxx.BIN EXEC data file has been loaded, the EMUQ.EXE loadable must be loaded into Concept in the **Loadables** dialog.

Loading Firmware for Quantum LL984 Hot Standby Mode

The Quantum CPUs not ending in X or S can be used for the LL984 Hot Standby mode. A special EXEC file must be downloaded onto the CPU for this. The loadable for LL984 Hot Standby (CHS_208.DAT) is automatically installed by the system.

Loading Firmware for Quantum IEC Hot Standby Mode

The 140 CPU 434 12 and 140 CPU 534 14 CPUs can also be used for IEC Hot Standby. A special EXEC file must be downloaded onto the CPU for this. The loadables for IEC Hot Standby (IHSB196.EXE and CHS_208.DAT) are automatically installed by the system.

Loading Firmware for Quantum Equation Editor

The Quantum CPUs not ending in X or S can be used for the LL984 equation editor. A special EXEC file must be downloaded onto the CPU flash for this. This EXEC file is not part of the Concept delivery range but can be obtained over the Internet at www.schneiderautomation.com.

Loading Firmware for Momentum PLC Type

The following table shows the current EXEC versions, which are located on the Service Release CD and supplied with Concept.

Momentum PLC type (CPU 171 CBB 970 30):

171 CBB	MPSV100.BIN (LL984 only)	MPSV100e.BIN (IEC only)	IEC memory (kbyte)
970 30-984	X	-	
970 30-IEC	-	X	236

Momentum PLC type (CPU 171 CCC 7x0 x0):

171 CCC	M1LLVxxx (LL984 only)	M1IVxxxE (IEC only)	IEC memory (kbyte)
760 10-984	X	-	
760 10-IEC	-	X	220
780 10-984	X	-	
780 10-IEC	-	X	220

Momentum PLC type (CPU 171 CCC 9x0 x0):

171 CCC	M1EVxxx (LL984 only)	M1EVxxxE (IEC only)	IEC memory (kbyte)
960 20-984	X	-	
960 30-984	X	-	
960 30-IEC	-	X	236
980 20-984	X	-	
980 30-984	X	-	
980 30-IEC	-	X	236

Momentum PLC type (CPU 171 CCS 7x0 x0):

171 CCS	M1LLVxxx (LL984 only)	M1IVxxxE (IEC only)	IEC memory (kbyte)
700 10	X	-	
700/780 00	X	-	
760 00-984	X	-	
760 00-IEC	-	X	160

The stripped EXEC of the M1 supports up to a maximum of 44 I/O modules.

Loading Firmware for Compact PLC Types

The **CTSXxxxD.BIN** EXEC file must be downloaded onto the CPU flash for all Compact CPUs.

Loading Firmware for Atrium PLC Types

A special EXEC file (see table below) must be downloaded onto the CPU flash for each Atrium CPU.

180 CCO	EXEC file
121 01	AI3Vxxxx.BIN
241 01 241 11	AI5Vxxxx.BIN AI5Vxxxx.BIN

PLC memory mapping

At a Glance

For the creation of the program, sufficient address zones for the necessary number of input bits, output/flag bits, input words and output/flag words are to be entered.

An overview of the state RAM value is also given:

- Max. state RAM
- State RAM in use
- State RAM use

An unassociated value is shown with an error message, and can be automatically suited to the given value.

IEC Hot Standby data

After configuration of an IEC Hot Standby system, enter sufficient address zones for the required number of input words. The higher the number of IEC Hot Standby input words, the larger the transmit buffers for the IEC component. This means all the bigger the IEC application in use can be.

⚠ CAUTION

System cycle time influence!

The size of the configured state RAM in an IEC Hot Standby project has a significant effect on the system cycle time. As soon as a configured cycle ends, the next starts after the transfer of all state RAM data to the CHS module.

Failure to follow these instructions can result in injury or equipment damage.

Loadables

Introduction

Loadables are loadable programs, which are only loaded into the PLC when required.

The various uses of loadables are described in the following sections.

Note: When you delete an Expert module in online mode in the I/O map the allocated loadable is also automatically deleted. If you wish to place this module back in the I/O map at a later time, it will be necessary to download again.

Downloading Loadables for the IEC Runtime System

The following loadables for the combined execution of IEC and LL984 programs (CPU 113 0x, CPU 213 0x or CPU 424 02) are available:

If...	Then...
you want to use CPUs with the mathematics processor for IEC programming,	install the loadable pairing @1S7196 and @2I7196.
you want to use CPUs without the mathematics processor for IEC programming,	install the loadable pairing @1SE196 and @2IE196.

Downloading Loadables for Expert Modules

The following loadables are available for Expert modules:

If...	Then...
you are configuring the 140 ESI 062 00 module with 32 bit runtime system and the 140-NOA-611-x0 module	install the loadable ASUP196. **Note:** The ULEX196 loadable is automatically installed. The ASUP 196 loadable is only installed automatically on 32-bit CPUs. On 16-bit CPUs with Stripped EXEC (QIECVxxx.BIN), the ASUP196 loadable must be installed afterwards.
you are configuring the 140 ESI 062 10 module,	install the loadable pairing NSUP + ESI. **Note:** These two loadables do not come with the Concept software package, but are supplied with the 140 ESI 062 10 module and must be unpacked at the time of installation (**Unpack...**).

Downloading Loadables for LL984

These are not included in the Concept delivery range. You can order these loadables via the "Automation Customer Service Bulletin Board (BBS)" (related topics README).

Downloading Loadables for Hot Standby

The following loadables for Hot Standby mode are available:

If...	Then
you are using the LL984 Hot Standby mode,	the loadable CHS_208 is automatically installed.
you are using the IEC Hot Standby mode,	the loadables IHSB196 and CHS_208 will be loaded automatically.

Downloading User Loadables

Loadables that are created by the user are called user loadables (*.EXE, *.DAT). They are located in the Concept directory DAT and using the **Unpack...** command button they can be inserted into the **Loadables** dialog at installation.

Downloading Loadables for IEC Support Only

The following loadables for IEC support only (CPU 113 xxS without mathematics processor) are available:

If...	Then
your application uses REAL arithmetic,	install the loadable EMUQ196. **Note:** The loadable is installed together with the EXEC-file QIECVxxx (installation in Concept EXECLoader).

Downloading Loadables for INTERBUS and IEC Support Only

The following loadables for IEC support are available:

If the CPU	Then
● 113 02S ● 113 03S ● 213 04S ● 534 14 ● 434 12 is configured,	install the loadable ASUP196. **Note:** The ULEX196 loadable is automatically installed. The ASUP 196 loadable is only installed automatically on 32-bit CPUs. On 16-bit CPUs with Stripped EXEC (QIECVxxx.BIN), the ASUP196 loadable must be installed afterwards.
113 03 is configured	install the loadable pairing @1SE196 + @2IE196. The ULEX196 loadable is automatically installed.
213 04 is configured,	install the loadable pairing @1S7196 + @2I7196. The ULEX196 loadable is automatically installed.

Downloading Loadables for INTERBUS and LL984 Support Only

The following loadables for LL984 support are available:

If the CPU	Then
● 113 02 ● 113 03 ● 213 04 is configured,	you can install the following loadables: ● ULEX196 ● @1S7196 + @2I7196 + ULEX196 **Note:** The ULEX196 loadable is automatically installed with this.
● 534 14 ● 434 12 is configured,	the loadables ASUP196 and ULEX196 will be loaded automatically.

Segment manager

At a Glance

If a remote I/O st. (Drop) is configured, the sequence and method of processing the LL984 section can be defined in the dialog box **Segment manager**.

When deleting (in the dialog box **I/O map**) a configured remote I/O st. (Drop), it is automatically deleted in the segment manager.

Mode of Functioning

Every I/O st. (Drop) is assigned a segment. It is therefore not permitted to enter fewer segments in the segment scheduler, than there are I/O st.s (Drops) configured in the I/O map. In the segment scheduler, the maximum segment numbers is by default set at 32.

The configurator checks the agreement between the two dialogs and classifies the I/O st.s (Drops) in the segment scheduler. A window informs you which I/O stations (Drops) have been inserted.

Altering the segment processing sequence

The sequence for segment processing can be altered manually, in that the segment number or I/O st. number can be edited in the corresponding line. For the local I/O st. (Drop), 1 is entered in the first line of the dialog box in the columns **In stat.** and **Out stat.** automatically.**1**

If no sequence was defined, the segments are processed in ascending order.

Sorting criteria for additional I/O st.s

Recently added I/O st.s (Drops) are classified in the segment manager according to the following criteria:

If…	Then…
A new I/O st. is added,	it is automatically classified behind the last available line.
All determined segments are already in use,	the last segment is reused for the input of the new I/O st. (Drop), i.e. a segment number can be repeated, as the stations are differentiated.

Available methods for segment processing

When setting the segment manager, the following methods of processing can be selected:

Processing type	Meaning
Continuous	Cyclic processing
Controlled	Manually controlled processing
WDT reset	Reset watchdog timer
End of logic	End of processing

Note: If subprograms are to be used in LL984, the last configured segment cannot be processed in the segment manager. The type of solution must unconditionally be **End of logic**.

Advanced settings in the segment manager

With the "Controlled" type of processing, only the reference numbers 0x and 1x are authorized, which determines when the logic for the corresponding section is processed.

The field **In. stat.** and **Out stat.** allow the input of corresponding I/O st. numbers, which must be configured. If a **0**is entered, no input/output is served by this segment number.

I/O Map

Introduction

In the I/O map, configure the I/O stations (drops) with the modules in use. Afterwards perform the I/O addressing and the parameterization of the configured modules.

Allocating Drops

Drop numbers can be allocated optionally except for the first one (from 2 to). The first drop number is automatically recognized as the local drop, and cannot be edited.

Configuring the Backplane Expander

The 140 XBE 100 00 module is necessary to expand the backplane. By doing this you can connect a second backplane, and gain 13 extra slots. The 140 XBE 100 00 module is mounted in both backplanes and, in addition, requires an independent power supply (power supply unit).

Expanded backplanes are configured in Concept in the first drop using slots 2-1 to 2-16.

A more detailed description about the configuration of expanded backplanes with the 140 XBE 100 00 module is given in the chapter *Backplane Expander Config, p. 117*.

⚠ **CAUTION**

The slot assignment of the 140 XBE 100 00 is not shown in the configurator, so a double assignment is possible.

You should take note of the hardware slots of the module and the power supply, and should not occupy these slots with other modules in the I/O map.

Failure to follow these instructions can result in injury or equipment damage.

Note: The flow of data via an expanded backplane is quicker than via the remote system.

Allocating the I/O Ranges

When allocating the I/O ranges the following references are allowed:

- 3x references for analog input modules
- 4x references for analog output modules
- 3x or 1x references for digital input modules
- 4x or 0x references for digital output modules
- 1x or 3x references for Expert modules (input)
- 0x or 4x references for Expert modules (output)

Note: The unique addressing is checked so that no addresses are occupied twice within the configuration.

Parameterization

Configured modules can be individually parameterized to determine the variable process conditioned settings.

Connection to other Network Systems

In addition to local and remote drops, links to other network systems can be established with configured coupling modules:

- Ethernet
- INTERBUS
- Profibus DP

See also the chapter entitled *Configuration of various network systems, p. 121* and *Configuration examples, p. 887*.

Read in Map

In the ONLINE mode of the stopped PLC, the hardware modules are listed in the I/O map and can be transferred as follows:

Step	Action
1	Open a project.
2	Open the **PLC Configuration** window.
3	Using the **PLC Type** menu command, open the **PLC Type** dialog and select the PLC type.
4	Connect the host computer to the PLC (**Online** → **Connect...**).
5	Open the **I/O Map** dialog (**PLC Configuration** → **I/O Map**).
6	Use the **Edit** command button to open the **Local Quantum I/O station** dialog.
7	Check the **Poll** check box. **Response:** The recognized modules are listed in the **Read** column in color.
8	Double click on the colored text boxes in the **Read** column. **Response:** The listed modules are transferred to the **Module** column.
9	Enter the address zone in the corresponding columns (**In.Ref.**, **In End**, **Out Ref.**, **Out End**).
10	After the hardware matching between the host computer and the PLC, the configuration can continue.

5.4 Optional configuration

At a Glance

Overview

This section contains the description of the optional configuration.

What's in this Section?

This section contains the following topics:

Topic	Page
Settings for ASCII Messages	106
Making Additional Functions Available in the Configurator	107
Data Exchange between Nodes on the Modbus Plus Network	108
How many words are really used when data is received (Peer Cop)	109
Protecting Data in the State RAM before Access	111
Parameterize interfaces	112
Special Options	114

Settings for ASCII Messages

Introduction

To create the ASCII messages, you are required first of all to set a mask, which contains the number of messages, the message area size and the ASCII ports. Once you have done that you can create the ASCII messages, which are then processed with the Ladder Logic programming language.

Precondition

ASCII messages are only possible in the Quantum family, and can only be processed with the LL984 processing language.

Procedure

To create the ASCII messages, you must first set the mask:

Step	Action
1	In the **PLC Configuration** → **ASCII** window, open the **ASCII Setup** dialog.
2	In the **Total Messages** text box specify a value from 1 to 999.
3	In the **Message Area Size** text box specify a value from 1 to 9999 bytes.
4	In the **ASCII Ports** text box specify an interface from 2 to 32.
5	Confirm your entries with the **OK** command button. **Response:** The settings are saved and the dialog is exited.
6	In the **Project** main menu open the **ASCII Message Editor** dialog (with the **ASCII Messages...** menu command).
7	Create the ASCII messages here, see also the description *ASCII Message Editor, p. 605*.

Making Additional Functions Available in the Configurator

Introduction

Additional functions can be used for the configuration, if they have previously been enabled or set in the **Select Extensions** dialog.

Activating Advanced functions/ dialogs

By checking the check box or setting the Ethernet modules the corresponding menu commands are enabled and can be edited in the **PLC Configuration** → **ASCII** window.

The following functions/dialogs can be activated:

- Data protection
- Peer Cop
- Hot Standby
- Ethernet I/O-Scanner

> **Note:** The available functions are dependent upon the configured CPU. Also see the online help "Select Extensions".

Specify Coupling Modules

Coupling modules must be configured in order to connect to other network systems. To do this, specify the number of modules in the corresponding list box, which are then available in the I/O map.

The following systems can be configured:

- TCP/IP Ethernet
- Symax-Ethernet
- MMS-Ethernet
- Profibus DP

> **Note:** The maximum number of coupling modules depends upon the configured CPU. Also see the online help "Select Extensions".

Data Exchange between Nodes on the Modbus Plus Network

Introduction

With a Modbus Plus (MB+) connection you can configure a PLC using the Peer Cop functionality, so that data exchange with another PLC is possible. In such a case, the Peer Cop takes data from a reference area within a "source" PLC and places this via the Modbus Plus (MB+) network into a determined reference range of a "destination" PLC. This operation is performed in the same identical way for each token rotation.

Using the Peer Processor, input data from other nodes on the local network can be received by the user program. Likewise, output data from the user program can be transmitted to other nodes on the local network.

The Peer Cop has two variants for data exchange:

- global data exchange
- specific data exchange

Precondition

The **Peer Cop** menu command is only available if, in the **Select extensions** dialog the **Peer Cop** check box is checked.

Global Data Exchange

With global data exchange, the data sent from the current "source" PLC is received by all "destination" PLC devices in the Modbus Plus (MB+) network. Up to 64 destination devices can be reached in this way, which can each receive the data in 8 destination addresses of the State RAM.

See also section "*How many words are really used when data is received (Peer Cop), p. 109*".

Specific Data Exchange

With specific data exchange, data is sent from a selected "source" PLC to a selected "destination" PLC in the Modbus Plus (MB+) network. To do this, enter the respective addresses for the data exchange in a table at the corresponding source and destination nodes (1-64).

The address must correspond to the MB+ node address on the back of the respective module. This address setting can be altered and must be specified before mapping. (See also hardware description)

Select the node to be read or written according to the hardware configuration.

How many words are really used when data is received (Peer Cop)

Introduction

The number of words used may not exceed 500. To avoid this a simple formula can be used, how many words are used on receipt.

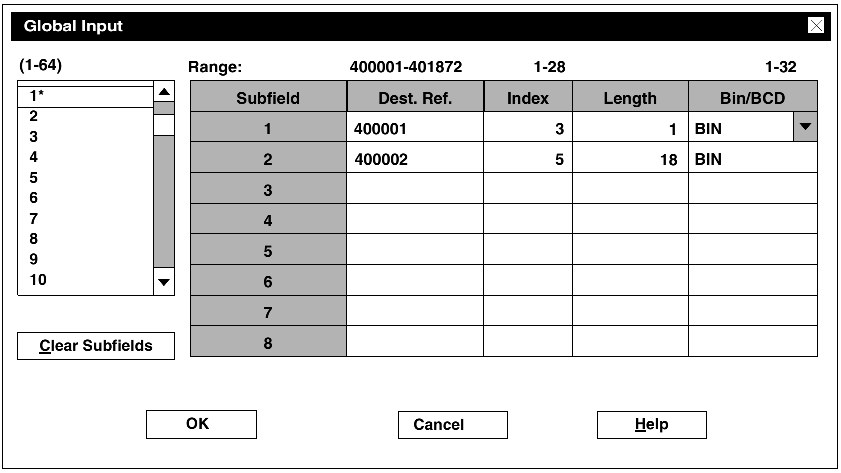
Formula

The formula, to find the number of words used is as follows:

Length + (index – 1) = number of words

Example

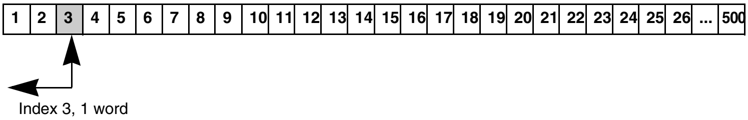
The Peer Cop dialog **Global Input** has the following entry:

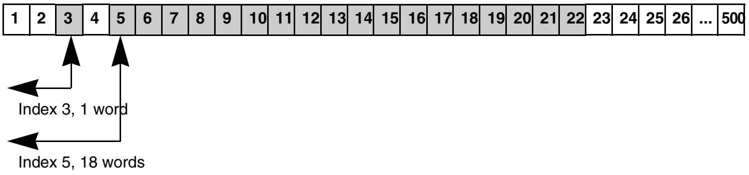


Range: 400001-401872 1-28 1-32

Subfield	Dest. Ref.	Index	Length	Bin/BCD
1	400001	3	1	BIN
2	400002	5	18	BIN
3				
4				
5				
6				
7				
8				

The following process takes place:

Step	Action
1.	Bus node 1 sends 1 word to the subfield start reference 400001, starting at index 3.
2.	At index 3 (word 3) the receipt of the data begins. (The preceding words are also counted.) Word 1 - 500  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 ... 500 Index 3, 1 word
3.	In total 3 words are required by subfield 1. Formula: 1 + (3 - 1) = 3
4.	Bus node 1 sends 18 words to the subfield start reference 400002, starting at index 5.

Step	Action
5.	At index 5 (word 5) the receipt of the data begins. (The preceding words are also counted.) Word 1 - 500  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 ... 500 Index 3, 1 word Index 5, 18 words
6.	In total 22 words are required by subfield 2. Formula: 18 + (5 - 1) = 22

Note: Only the largest number of words used per bus node by be taken into account. In the example 22 words from a maximum of 500 permitted words are used.
For more bus nodes the largest number of words used per bus node must be added.
For example:

Bus node 1 with 22 words
+
Bus node 2 with 28 words

50 words from 500 words allowed.

Protecting Data in the State RAM before Access

Introduction

Output address ranges (coils and registers) can be protected by specifying the address from which writing is possible in the **Data Protection** dialog. All addresses before this are write-protected.

Precondition

The **Data Protection** menu command is only available if, in the **Select Extensions** dialog, the **Data Protection** check box is checked.

Entering Access Protection

This access protection operates in connection with "normal" data access, which happens externally via a Modbus or Modbus Plus interface. Access from the host computer out is in any case permitted and bypasses this protection mechanism.

Parameterize interfaces

At a Glance

Depending on their use in Concept, the following interfaces must be parameterized:

- ASCII interface
- Modbus interface

Parameterize ASCII interface

For an ASCII message transmission, the serial communication parameters for the port interfaces can be specified in the **ASCII port settings** dialog box.

Note: The **ASCII port settings** dialog box is only available when the number of ASCII ports has been specified beforehand in the dialog box **ASCII set up**.

Parameterize Modbus interface

For a Modbus coupling, in the dialog box **Modbus port settings** the serial communication parameters of the port interface can be entered on the programming device, on a CPU and the NOM assemblies (Network Option Module).

⚠ CAUTION

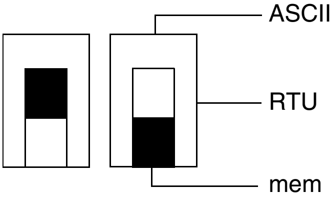
Do not make any online changes since this will cause all Editors to close!

The Modbus port settings should not be altered in Online mode, or else all Editors are automatically closed.

Failure to follow these instructions can result in injury or equipment damage.

Note: The settings for a Modbus coupling in Concept only have an effect if the switch on the front of the assembly is at the lowest position (mem).

Switch position on the NOM



Note: If the left-hand switch is in the upper position and right-hand switch is set to mem then, as of firmware version 2.20, bridge mode is deactivated. This means that the network connection between Modbus and Modbus Plus is locked.

Interface parameterization with network connections between Modbus and Modbus Plus

A network connection between Modbus and Modbus Plus nodes can be made in the dialog box **Modbus port settings** by checking the check box **Bridge mode**.

Note: The settings are only effective if the switch on the front of the assembly is in the middle position (RTU).

Special Options

Introduction

In the **Specials** dialog you can configure special options:

- Battery coil
- Timer register:
- Time stamp for MMI applications (TOD)
- Allow duplicate coils
- Watchdog-Timeout (ms)
- Time slice for online changes (ms)

Battery coil

You can specify an address of a coil, which shows the status of the battery. This assignment is used for battery monitoring. In this way, the weak battery can be replaced early to avoid a loss of data.

Timer Register:

The content of the time register is incremented every 10 ms and has a free value between 0000 and FFFF hex.

Time for MMI applications (Date/Time)

This time stamp is only intended for a MMI application. Eight registers are reserved for setting the clock.

The TOD input (Time of Day) is in the American format:

4xxxx	**Control register**	
	Discrete 1 (MSB) Discrete 2 Discrete 3 Discrete 4	1 = set clock values 1 = read clock values 1 = preset discrete 1 = error discrete
4xxxx+1	Day of week (1 - 7)	
4xxxx+2	Month (1 - 12)	
4xxxx+3	Day (1 - 31)	
4xxxx+4	Year (00 - 99)	
4xxxx+5	Hours (0 - 23)	
4xxxx+6	Minutes (0 - 59)	
4xxxx+7	Seconds (0 - 59)	

Allow Duplicate Coils

You can assign several outputs to a coil. To do this, check the check box, and specify the first address to which several outputs can be allocated in the **First Coil Address:** text box.

Note: This function is unavailable with the Momentum PLC family.

Watchdog Timeout (ms*10)

You can set a pulse supervision for the user program by entering a numerical value of between 2 and 255 (ms). As soon as there are no count pulses within the specified time, an error message will appear.

Time Slice for Online Changes (ms)

You can set a time supervision for the communication between the nodes by entering a numerical value between 3 and 30 (ms). As soon as there is no communication within the specified time, an error message will appear.

5.5 Backplane Expander Config

At a glance

Introduction

This chapter describes the function and configuration of the backplane expander.

What's in this Section?

This section contains the following topics:

Topic	Page
Generals to Backplane Expander	118
Edit I/O Map	119
Error handling	120

Generals to Backplane Expander

Introduction

The Quantum backplane expander provides a single backplane expansion to a local drop or a RIO drop through the 140 XBE 100 00 module.

Function description

The module connects two Quantum backplanes (primary and secondary) through a custom cable and support all data communication between the backplanes. Each backplane requires a 140XBE10000 module that occupy a single slot and requires its own power supply.

Procedure at an Error

The backplane expander is designed in the way that if it is not installed or improperly connected, it will not effect the functionality of the primary rack. Only the backplane expander installed and connected properly, the both racks are then able to communicate and controlled by prime CPU or RIO drop controller.

Edit I/O Map

Requirements

Currently only Quantum controllers support backplane expander. Primary rack contains the CPU or RIO drop controller and is allowed to config all type of additional modules up to the physical slot address limitation. All I/O modules can be also added to the secondary rack. However, option modules, such as NOMs, NOEs and CHSs must reside in the primary rack.

To place a module in proper rack, it is necessary to add an extra attribute in the I/O module database to specify that the module is available only for the primary or secondary or both.

Configuration in I/O Map

Exist Quantum local drop or RIO drop only support one rack up to sixteen slots. With backplane expander, it is extended as if the drop support two racks, and each has sixteen slots. By clicking at the button **...** on **Module** column, all modules available to the rack clicked (primary or secondary) will show in the module selection dialog that can be selected and assigned to the current slot.

Each rack requires a 140 XBE 100 00 module to make backplane expander work properly.

Note: The 140 XBE 100 00 module does not have a personality code and therefore can not be recognized by the Concept.

The module will just look like an unfilled slot in the Concept I/O map. If any module is configured in the secondary rack, it is user’s responsibility to ensure there is one slot in each rack that is reserved for 140 XBE 100 00 module and all hardware are connected properly.

Error handling

Introduction

The validate processes for the primary rack will be applied to the secondary rack too, such as duplicate reference, missing input or output reference, etc. Besides existing regular validation, traffic cop will do some special check for the backplane expander.

No reserved slot for 140 XBE 1000 00

If any module is found in the secondary rack and there is no empty slot left in either of racks when user trying to exit the rack editor dialog, an error message will be displayed: "There must be one empty slot reserved for 140 XBE 100 00 module in each rack to make backplane expander work." The rack editor dialog will then not be closed.

Special module in secondary rack

To prevent any special module (such as, NOE, CHS, etc) being added to the secondary rack, rack editor dialog do not allow to cut/copy these head modules. It will also check module personalities before user try to do any paste operation. If some unsupported module for the secondary rack is found, an error message will be displayed: "The buffer contains some module that can not reside in the secondary rack." The paste operation will be aborted.

5.6 Configuration of various network systems

At a Glance

Overview

This section contains the description of the configuration of various network systems.

What's in this Section?

This section contains the following topics:

Topic	Page
Configure INTERBUS system	122
Configure Profibus DP System	123
Configure Ethernet	125
RTU extension	127
Ethernet I/O Scanner	128
How to use the Ethernet / I/O Scanner	131

Configure INTERBUS system

At a Glance

The configuration of the INTERBUS system can take place within the PLC families of Quantum and Atrium.

INTERBUS configuration with Quantum

With the Quantum family the coupling of a remote bus takes place in a Quantum I/O station (Drop). To do this, the INTERBUS Master NOA 611 00 must be configured and parameterized in the CMD tool (Configuration Monitoring and Diagnostic Tool).

See also Configuration example 4 (see *Quantum Example – INTERBUS Control, p. 919*).

INTERBUS configuration with Atrium

With the Atrium family, the coupling of the remote bus takes place via the master assembly 180 CCO 121 01, 180 CCO 241 01 or 180 CCO 241 11 in this way, the INTERBUS Master CRP 660 0x is automatically inserted into the local I/O station (Drop). The INTERBUS I/O station (Drop) nodes are configured in the CMD tool (Configuration Monitoring and Diagnostic tool), saved as a *.SVC data file and imported to Concept. After the import into the I/O map the configuration can be changed afterwards in Concept.

See also Configuration example 9 (see *Atrium Example – INTERBUS Controller, p. 963*).

Configure Profibus DP System

Introduction

The configuration of the Profibus DP system can take place within the PLC families of Quantum and Atrium.

Profibus DP Configuration with Quantum

With the Quantum family the connection to the Profibus DP system takes place in a Quantum drop. To do this, you must first of all set the number of bus controllers (CRP 811 00) used in the **Select Extensions** dialog. The modules then appear in the list box of the **I/O Module Selection** dialog and can be inserted into the I/O map.

The configuration of the Profibus DP node is created in the SyCon configuration tool, saved as a *.CNF file and transferred directly to Concept. However, the configuration (*.CNF) can be imported to Concept at a later time.

⚠ CAUTION

PROFIBUS DP ADDRESSES MAY BE OVERWRITTEN

When working with Profibus DP configuration make sure that the addresses of two 8 bit E/A modules without gap to the following 16 bit limit is only permitted when both 8 bit modules belong to the same Profibus DP master. If you do not adhere to this guideline, the input bits of one module (e.g. Profibus DP Master A) may be overwritten by the other module (e.g. Profibus DP Master B).

Failure to follow these instructions can result in injury or equipment damage.

Importing the Profibus DP Configuration

To import the configuration (*.CNF) to Concept, proceed as follows:

Step	Action
1	In the **PLC Configuration** window, open the **I/O Map** dialog.
2	Select the drop and use the **Edit**dialog **Local Quantum I/O Drop**.
3	Double click on the in the **Module**column. **Reaction:** The **I/O Modules Selection** dialog is opened.
4	In the **I/O Adapter** column, select the CRP-811-00 module, and press the **OK** command button. **Reaction:** The CRP-811-00 will be inserted in the I/O map.
5	In the **Local Quantum I/O Drop** dialog, select the line of the mapped bus controller (CRP-811-00) and press the **Params** command button. **Reaction:** The **CRP-811-00 (Profibus DP)** dialog will open.
6	Using the **Import** open the **Select Import File** window.
7	To import, specify the path of the CNF file, and press the **OK** command button. **Reaction:** The Profibus DP configuration is entered in the Concept I/O map. **Note:** After the Profibus DP nodes are entered into Concept, the reference ranges for all modules and diagnostic data must be edited later.

Configuration Example

An example of configuration is given in Example 11 (see *Quantum Example - Profibus DP Controller, p. 933*).

Configure Ethernet

Introduction

An Ethernet bus system can be configured within the following PLC families:

- Quantum
- Atrium
- Momentum

Precondition

In order to connect to the Ethernet bus system, a PCI network card must be available in the host computer. Afterwards the Ethernet interface needs to be parameterized and the drivers that are provided on CD need to be installed (*Configure Ethernet, p. 986*).

After the Ethernet module has been slotted into the central backplane, the internet address, subnet mask, gateway and frame type can be allocated by the network administrator.

Configuration with Quantum

The procedure for Ethernet configuration in Concept is as follows:

Step	Action
1	In the **PLC Configuration** window, open the **Select Extensions** dialog.
2	Enter the number of Ethernet modules (NOE) in the text boxes. **Response:** The modules then appear in the list box in the **I/O Module Selection** dialog and can be inserted into the I/O map.
3	In the **PLC Configuration** window, open the **Ethernet I/O Scanner**dialog, in which you enter the information from the network administrator (Internet address, subnet mask, gateway, frame type).
4	In the **Online** main menu, open the **Connect to PLC** dialog (menu command **Connect...**).
5	In the **Protocol Type** list box, select the option **TCP/IP**, and in the **IP address or DNS Hostname** text box, enter the address of the TCP/IP card.
6	After programming, in the **Online** main menu, open the **Load into PLC** dialog (menu command **Load...**), and click on the **Load** command button. **Response:** A message appears, asking whether you would like to start the PLC.
7	Before you confirm the message with the **Yes** command button, the display "link" must appear on the Ethernet module.

Error Action

After configuration, only start the PLC once the display "link" has appeared on the Ethernet module. If this is not the case, withdraw the Ethernet module from the central backplane and then slot it in again. If the display "link" is still not shown, there must be a serious error.

Available Ethernet Modules

The maximum number of NOE modules is dependent upon the configured CPU (select in the **PLC Selection** dialog):

CPUs	Number of NOE modules
113 02/S/X	0 - 2
113 03/S/X	0 - 2
213 04/S/X	0 - 2
424 0x/X	0 - 6
434 12	0 - 6
534 14	0 - 6

Configuration with Momentum

The configuration of the Ethernet bus system with Momentum is described in the section *Momentum Example - Ethernet Bus System, p. 985*.

RTU extension

Requirements

To make the RTU menu command available you have to choose a Compact CPU with LL984 programming language in the **PLC Selection** dialog.

CTS-/RTS-Delay

In this dialog you can set time delay for CTS or RTS independently for Comm port 1 of your Compact PLC. This feature allows modem communications with radios that require longer time frames. The delay time range is 0 ... 500 ms using 10 ms units. Enter the time delays your require.

Secured Data Area (SDA)

This feature allows you to configure an area in RAM that is secured from being overwritten. Secured Data Area (SDA) is a block of the Compact PLCs RAM that is set aside as 6x data space. The SDA can only be written to by specific functions that require secured data storage. General purpose Modbus commands, builtins, can not write to the SDA. Modbus Read (function 20) is able to read from the SDA, Modbus Write (function 21) is not able to write to the SDA. The SDA size range is 0 ... 128 K words using only 1 K word blocks. Enter the size your require.

Refer to the applicable user manual for the specific function for the required SDA size. For example, for Gas Flow, refer to the "Starling Associates Gas Flow Loadable Function Block" User Guide (890 USE 137 00).

PLC Login Password Protection

For the description of password protection, refer to section *Set PLC Password, p. 658*.

Ethernet I/O Scanner

Introduction

This function is for the following Quantum modules available:

- 140-NOE-211-x0
- 140-NOE-251-x0
- 140-NOE-771-xx

This function is for the following Momentum modules available:

- 171-CBB-970-30
- 171-CCC-960-20
- 171-CCC-980-20
- 171-CCC-980-30
- 171-CCC-960-30

Ethernet address and I/O scanning parameters can be modified using the **Ethernet / I/O Scanner** dialog box. From the **PLC Configuration** window, select **Ethernet / I/O Scanner**. This menu option will only be available if you have selected an M1 Processor Adapter with an Ethernet port or have Quantum TCP/IP Ethternet modules (NOE) as specified above.

This section describes how to configure the Ethernet port, including IP address, other address parameters and I/O scanning.

Ethernet Configuration Options

The Ethernet / I/O Scanner screen offers three options for configuring the Ethernet port on an M1 Processor Adapter:

Configuration options	Meaning
Specify IP Address	This is the default option. It allows you to type the IP address, gateway and subnet mask in the text boxes in the upper righthand corner of the screen.
Use Bootp Server	Click this radio button if you want the address parameters to be assigned by a Bootp server. If you select this option, the address parameter text boxes in the upper righthand corner of the screen will be grayed out. They will not display the actual address parameters.
Disable Ethernet	Click this radio button if you want to disable the Ethernet port. Disabling the port will reduce the scan time for the Processor Adapter.

Setting Ethernet Address Parameters

If you choose to specify the IP address, you should complete all four text boxes in the upper righthand corner of the dialog box:

Parameters	Meaning
Internet Address	Type a valid IP address in the **Internet Address** text box (for example: 1.0.0.1). **Caution:** POTENTIAL FOR DUPLICATE ADDRESSES! Obtain a valid IP addresses from your system administrator to avoid duplication. **Failure to observe this precaution can result in injury or equipment damage.**
Gateway	Consult your system administrator to determine the appropriate gateway. Type it in the **Gateway** text box.
Subnet Mask	Consult your system administrator to obtain the appropriate subnet mask. Type it in the **Subnet Mask** text box (for example: 255.255.255.0).
Frame Type	For NOE there is an additional Frame Type field. Your two possible choices are ETHERNET II or IEEE 802.3

Configuring I/O

Once the Ethernet port address parameters have been set, you may assign parameters for I/O scanning.

The text box **Master Module (Slot)** contains the Module type that you have configured for Ethernet communications. In the case of the Momentum Ethernet controller the slot will always be number 1, and the configured module type is displayed in the variable dialog field. If you are configuring a NOE in a standard rack the slot number assigned in the I/O Map will be displayed along with the module type. Until the I/O Map is conmpeted this test field will indicate "Unassigned". In instances where more than one NOE is configured the I/O Scan parameters reflect the unit currently in the dialog box from which you can select the additional unit by activating the Pulldown list.

The text field **Health Block (1x/3x)** is only available by using the 140-NOE-771-xx. The health timeout is used for setting the health bit. If the response arrives before the end of the HealthTimeout period, the health bit is set; otherwise it is cleared. If the Health Timeout is zero, the health bit is set to true once communications are established, and it is never cleared.

Note: The configuration of the health block, refer to the user guide Quantum NOE 771 xx Ethernet Modules, model no. 840 USE 116 00.

The text box **Diagnostic Block (3x/4x)** is only available by using the Momentum Ethernet (M1E) and allows you to define the starting register of a number of bits which are used for diagnostic. The block can be specified in either 3x or 4x registers. For more information, refer to the user guide Quantum NOE 771 xx Ethernet Modules, model no. 840 USE 116 00.

I/O Scanner Configuration table:

Column	Description
Slave IP Address	Type the IP address of the slave module in this column (for example: 128.7.32.54). This address will be stored in a pulldown menu, so that you may use it in another row by clicking on the down arrow and selecting it.
Unit ID	If the slave module is an I/O device attached to the specified slave module, use the Unit ID column to indicate the device number. The Unit ID is used with the Modbus Plus to Ethernet bridge to route to Modbus Plus networks.
Health Timeout	Use this column to specify the length of time in ms to try the transaction before timing out. Valid values are 0 ... 50 000 ms (1 min). To avoid timing out, specify 0.
Rep Rate	Use this column to specify how often in ms to repeat the transaction. Valid values are 0 ... 50 000 ms (1 min). To repeat the transaction continually, specify 0.
Read Ref Master	Use the read function to read data from the slave to the master. This column specifies the first address to be read (for example: 400001).
Read Ref Slave	Use the read function to transfer data from the slave to the master. This column specifies the first address of up to 125 to read to (for example: 400050).
Read Length	Use the read function to read data from the slave to the master. This column specifies the number of registers to read (for example: 20).
Write Ref Master	Use the write function to write data from the master to the slave. This column specifies the first address to write (for example: 400100).
Write Ref Slave	Use the write function to write data from the master to the slave. This column specifies the first address of up to 100 to write to (for example: 400040).
Write Length	Use the write function to write data from the master to the slave. This column specifies the number of registers to write (for example: 40).
Description	You can type a brief description (up to 32 characters) of the transaction in tis column.

Note: You may include read and write commands on the same line.

How to use

For more information about how to use the Ethernet / I/O Scanner dialog see section *How to use the Ethernet / I/O Scanner, p. 131*.

How to use the Ethernet / I/O Scanner

Introduction

This section describes how to complete your Ethernet I/O configuration using the **Copy**, **Cut**, **Paste**, **Delete** and **Fill Down** buttons.

Copy and Paste

To save time when typing similar read and write commands, you may copy and paste entire rows within your configuration:

Step	Action
1	Select the row you want to copy by clicking on the row number at the far left.
2	Click the **Copy** button above the I/O configuration list.
3	Select the row where you would like to paste the data (by clicking on the row number at the far left).
4	Click the **Paste** button.

Cut and Paste

To move a row within the configuration list, follow the direction:

Step	Action
1	Select the row you want to move by clicking on the row number at the far left.
2	Click the **Cut** button above the I/O configuration list.
3	Select the row where you would like to paste the data (by clicking on the row number at the far left).
4	Click the **Paste** button. **Note:** Multiple rows may be cut/copy and pasted. The number of rows actually pasted is limited by the number of rows selected. For example if you copy 10 rows to the clipboard, then select an area of 6 rows to past, only the first six rows of clipboard data is pasted.

Delete

To delete a row within the configuration list, follow the direction:

Step	Action
1	Select the row you want to delete by clicking on the row number at the far left.
2	Click the **Delete** button above the I/O configuration list. **Note:** Multiple rows may be deleted.

Fill down

To copy part of any row to the next row or to a series of adjoining rows, use the **Fill Down** button, following the steps in the table

Step	Action
1	Use your mouse to select the data you would like to copy and the cells you would like to copy it to. **Note:** You must select one contiguous block of cells, with the data to be copied in the first row. You cannot select two separate blocks.
2	Click the **Fill down** Button. **Result:** The data from the first row is copied to the selected cells in the defined block.

NOE Ethernet modules

In this dialog the NOE Ethernet modules 140 NOE 211 x0,140 NOE 251 x0 and 140 NOE 771 10 are parameterized (in the **Ethernet Configuration** area).

In this dialog the NOE Ethernet module 140 NOE 771 00 is parameterized and addressed (in the **I/O Scanner Configuration** area).

For the followings modules you receive an function description:
- 140 NOE 211 x0
- 140 NOE 251 x0
- 140 NOE 771 xx

Momentum Ethernet modules

In this dialog the Momentum Ethernet modules are addressed (in the **I/O Scanner Configuration** area).

For the followings modules you receive an function description:
- 171 CBB 970 30 IEC
- 171 CBB 970 30 984
- 171 CCC 980 30 IEC
- 171 CCC 980 30 984
- 171 CCC 980 20 984
- 171 CCC 960 30 IEC
- 171 CCC 960 30 984
- 171 CCC 960 20 984

5.7 Quantum Security Settings in the Configurator

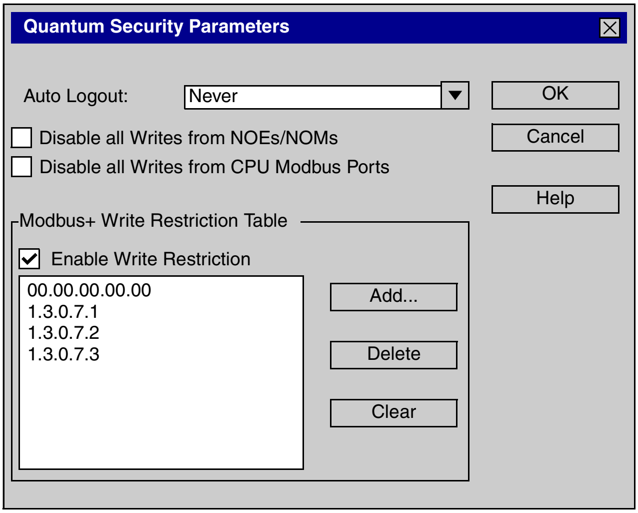
Quantum Security Parameters

Introduction

Various security parameters can be defined in the configuration of the Quantum CPUs 140 434 12A and 140 534 14A/B which are indicated in the log file *.LOG. This guarantees secure process documentation which includes the logging with the automatic logout, write access of NOEs/NOMs on the PLC as well as limited participants (max. 12) for network write access.

The definition of the security parameters can be found in dialog **Configuration** → **Security Expansion**.

Dialog **Quantum Security Parameters**:



Quantum Security Parameters

Auto Logout: Never

□ Disable all Writes from NOEs/NOMs

□ Disable all Writes from CPU Modbus Ports

Modbus+ Write Restriction Table

☑ Enable Write Restriction

00.00.00.00.00
1.3.0.7.1
1.3.0.7.2
1.3.0.7.3

Requirements

The security parameters are only available if the following conditions have been met:

- Supervisor Rights (see Concept under **Help** → **Info...** → **Current User:**)
- only with CPUs 140 CPU 434 12A and 140 CPU 534 14A/B

Automatic Logout

The automatic logout procedure logs a user out as soon as a predefined time limit (max. 90 minutes) is reached with no activity on the connection. This could be a lack of read or write activity from the programming device to the PLC for example.

The **Never** setting disables this function, i.e. automatic logout cannot occur.

Note: Automatic logout does not function if:
- the programming device (Concept) is connected to the PLC not via the local Modbus Plus Port of the CPU, but via a NOE/NOM module
  and
- another device is connected to the same NOE/NOM module with read access to the PLC.

Disable All Writes from NOEs/NOMs

By disabling all write accesses of

- 140 NBE 210 00 (ID-Code 0x0406)
- 140 NBE 250 00 (ID-Code 0x0407)
- 140 NOE 211 00 (ID-Code 0x0404)
- 140 NOE 251 00 (ID-Code 0x0405)
- 140 NOE 311 00 (ID-Code 0x0408)
- 140 NOE 351 00 (ID-Code 0x0409)
- 140 NOE 511 00 (ID-Code 0x040A)
- 140 NOE 551 00 (ID-Code 0x040B)
- 140 NOE 771 00 (ID-Code 0x040D)
- 140 NOE 771 01 (ID-Code 0x0422)
- 140 NOE 771 10 (ID-Code 0x040E)
- 140 NOE 771 11 (ID-Code 0x0423)
- 140 NOM 211 00 (ID-Code 0x010C)
- 140 NOM 212 00 (ID-Code 0x010C)
- 140 NOM 252 00 (ID-Code 0x010C)
- 140 NWM 100 00 (ID-Code 0x0420)

to the PLC, all write instructions are ignored by the CPU and responded to with an error message.

Note: MSTR read operations are not executed if the check box **Disable All Writes from NOEs/NOMs** is checked. (this also means the error state of the MSTR block shows no error!)

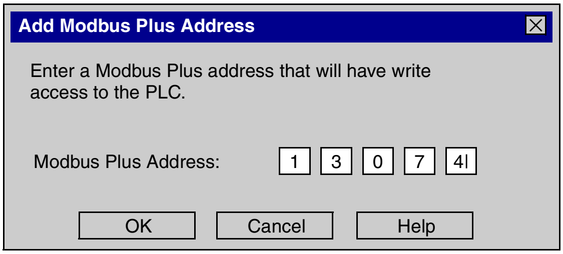
Disable all Writes from CPU Modbus Ports

To disable writes from the Quantum CPU Modbus connections, check the **Disable all Writes from CPUs from Modbus Ports** check box.

Limited Write Access on the Modbus Plus Network

A restricted number of participants that have access to the PLC can be configured for the Modbus Plus network. A maximum of 12 participants are allowed, the participant address of the programming device is automatically entered in the participant list and cannot be deleted.

Dialog **Add Modbus Plus Address** (press **Add...**)

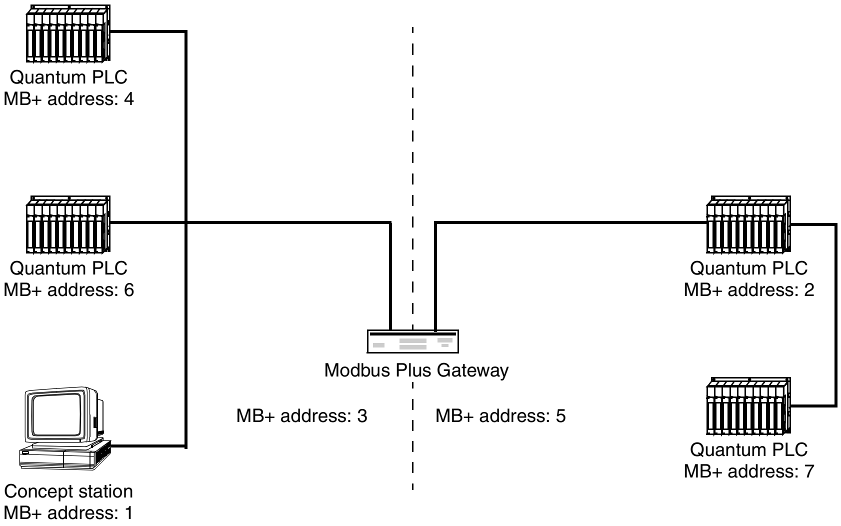


Enter a Modbus Plus address that will have write access to the PLC.

Modbus Plus Address: 1 3 0 7 4|

Examples of Modbus Plus paths

Modbus Plus network:



The address must be entered from the point of view of the receiving PLC to the sender, and thus begins at the first gateway or the next PLC. This depends on whether the sender and the receiver are located in the same Modbus Plus segment (no bridges/gateways) or whether the sender and the receiver are located in different segments (separated by one or more bridges/gateways).

Example 1:

Concept (MB+ Address 1) writes to PLC 6. There are no bridges or gateways between the two participants. Thus, the address entered appears as follows: 1 or 1.0.0.0.0

Example 2:

PLC 2 (MB+ Address 2) writes to PLC 6. A gateway (MB+ Address 3) is located between the two participants. Thus, the address entered appears as follows: 3.2.0.0.0

Note: Only the first Modbus Plus address can be recognized by the PLC. Thus, as soon as this first address is a bridge or a gateway, all devices in the network behind the bridge or gatway have write access to the PLC. Thus, in our example PLC 7 could also write to PLC 6 (Address: 3.7.0.0.0).

Main structure of PLC Memory and optimization of memory

6

At a Glance

Overview

This Chapter describes the main structure of the PLC Memory and the optimization of the memory with the different PLC families.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Main structure of the PLC Memory	139
6.2	General Information on Memory Optimization	141
6.3	Memory Optimization for Quantum CPU X13 0X and 424 02	147
6.4	Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)	161
6.5	Memory optimization for Compact CPUs	173
6.6	Memory optimization for Momentum CPUs	183
6.7	Memory optimization for Atrium CPUs	189

6.1 Main structure of the PLC Memory

General structure of the PLC Memory

At a Glance

In principle, the memory of a PLC consists of three parts:
- the memory for the Exec file,
- the state RAM and
- the program memory.

Memory for the EXEC file

The EXEC file contains the operating system and one or two runtime systems (IEC and/or LL984) for operating the user programs.

State RAM

The state RAM can be divided into different zones:
- the used 0x, 1x, 3x and 4x references,
- a reserve for further 0x, 1x, 3x and 4x references,
- possibly an extended memory zone for 6x references.

Program Memory

The program memory can be divided into different zones:
- the I/O map etc.,
- a reserve for extensions,
- the ASCII messages (if used), the Peer Cop configuration (if used), the Ethernet configuration (if used) etc.,
- a reserve for extensions,
- the IEC loadables (if required),
- the Global Data, consisting of the Unlocated Variables,
- the IEC program memory with the program codes, EFB-Codes and program data (section data and DFB instance data),
- possibly the ULEX loadable for INTERBUS or other loadables,
- the LL984 program memory.

6.2 General Information on Memory Optimization

Introduction

Overview

This Section contains general information on memory optimization.

What's in this Section?

This section contains the following topics:

Topic	Page
Possibilities for Memory Optimization	142
PLC-Independent	143

Possibilities for Memory Optimization

Description

The possibilities for memory optimization are partly dependent on the PLC family and CPU used:

- *PLC-Independent, p. 143*
- *Memory Optimization for Quantum CPU X13 0X and 424 02, p. 147*
- *Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B), p. 161*
- *Memory optimization for Compact CPUs, p. 173*
- *Memory optimization for Momentum CPUs, p. 183*
- *Memory optimization for Atrium CPUs, p. 189*

PLC-Independent

Introduction

There are 3 PLC-independent possibilities for memory optimization:

- *Optimize State RAM for 0x and 1x References, p. 144*
- *Only Download Required Loadables, p. 145*
- *Optimize Expansion Size, p. 146*

Optimize State RAM for 0x and 1x References

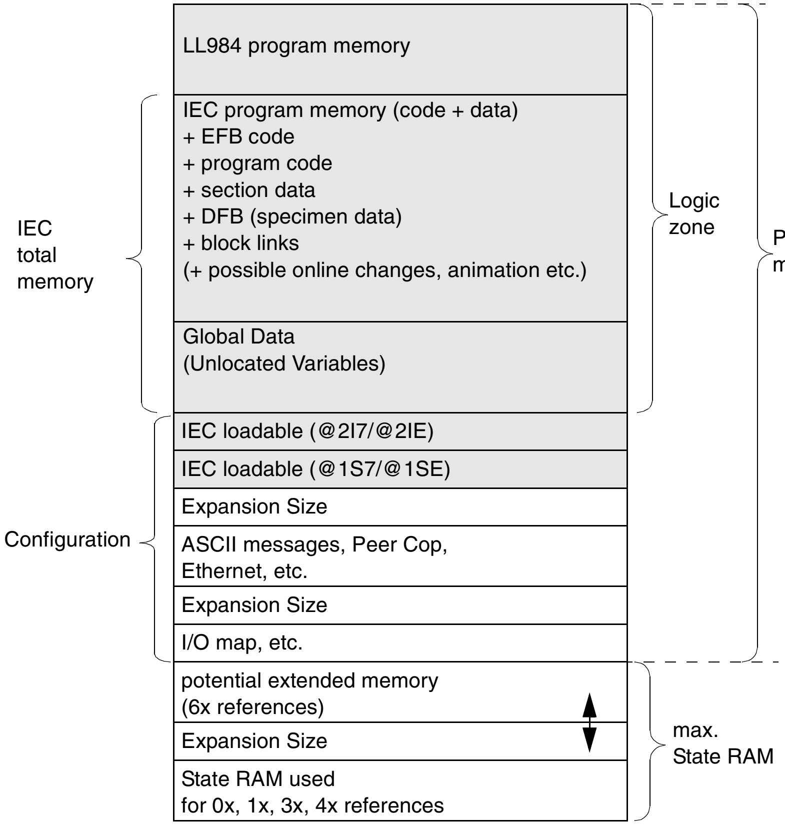
The state RAM contains the current values of the 0x, 1x, 3x and 4x references.

Even if the state RAM zone is outside the program memory zone, the size of the state RAM for 0x and 1x references influences the size of the program memory. Therefore, do not select a state RAM zone that is too large. In theory, the procedure only needs as many 0x and 1x references as the hardware requires. However, you will require a somewhat larger number of references if the I/O map is to be extended. It is advisable to be generous with the number of references during the creation phase of the user program when frequent changes are still being made. At the end of the programming phase, the number of these references can be reduced in order to create more space for the user program.

The settings for the 0x-, 1x-references can be found in **Project** → **PLC Configurator** → **PLC Memory Partition**.

In this dialog box, there is an overview of the size of the occupied state RAM zone and the percentage of the maximum state RAM that this represents.

Optimize state RAM for 0x, 1x, 3x and 4x references:



Only Download Required Loadables

All the installed loadables are downloaded into the program memory zone and occupy space. Therefore, only install those loadables which you really need (related topics *Loadables, p. 96*).

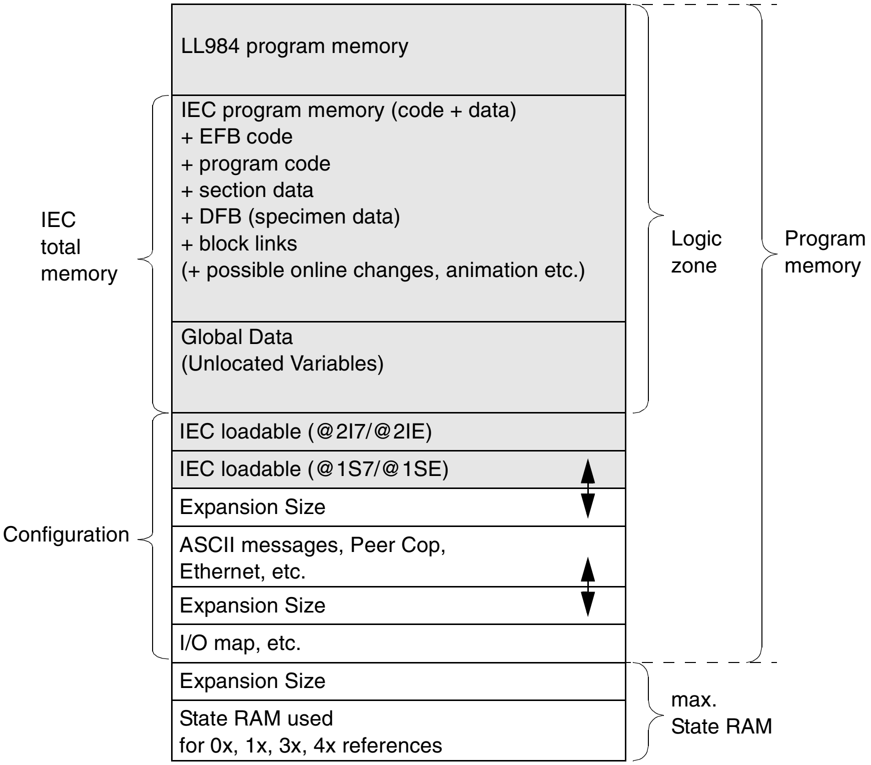
The memory space occupied by the installed loadables is displayed in the **Loadables** dialog box under **Used Bytes** (**Project** → **PLC configurator**). This information is calculated from the size of the loadable files and from the memory size assigned to the loadables.

Optimize Expansion Size

Each time, there is the possibility to reserve memory space for later expansion in the mapping zone (I/O map) and in the configuration expansion zone (Peer Cop). This memory space is necessary if e.g. the I/O map or the Peer Cop settings should be changed online. It is advisable to overestimate the reserves during the installation phase of the user program, that is, when modifications are often being made. At the end of the programming phase the reserves may be reduced again, to provide more space for the user program.

The settings for the mapping reserves are found in **Project** → **PLC Configurator** → **I/O Map** → **Expansion Size**. The settings for the Peer Cop reserves can be found in **Project** → **PLC Configurator** → **Config. Extensions** → **Select Extensions** → **Peer Cop** → **Expansion Size**.

Optimize Expansion Size



6.3 Memory Optimization for Quantum CPU X13 0X and 424 02

Introduction

Overview

This Section describes the memory optimization for the Quantum CPUs CPU X13 0X and CPU 424 02.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Quantum CPU X13 0X and 424 02	148
Selecting Optimal EXEC File	150
Using the Extended Memory (State RAM for 6x references)	154
Harmonizing the IEC Zone and LL984 Zone	156
Harmonizing the Zones for Global Data and IEC Program Memory	158

General Information on Memory Optimization for Quantum CPU X13 0X and 424 02

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project** → **PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

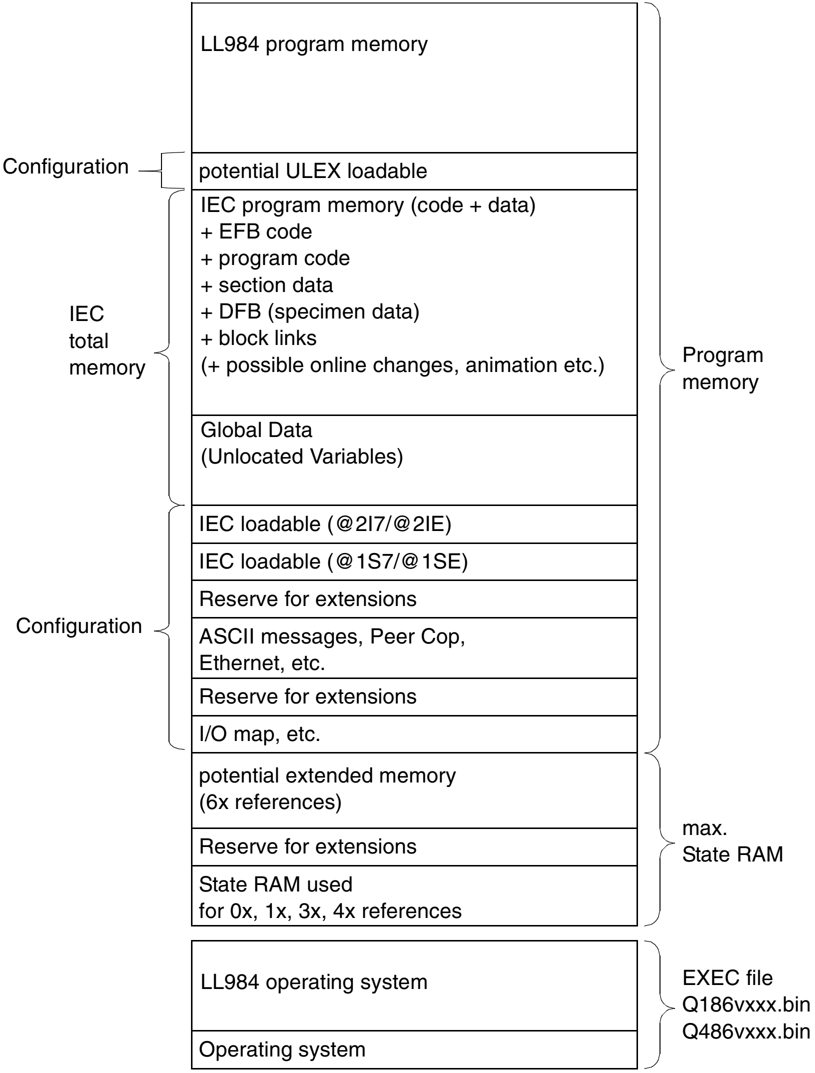
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Selecting Optimal EXEC File, p. 150*
- *Using the Extended Memory (State RAM for 6x references), p. 154*
- *Harmonizing the IEC Zone and LL984 Zone, p. 156*
- *Harmonizing the IEC Zone and LL984 Zone, p. 156*

Note: Also note the PLC-independent possibilities for memory optimization (see *General Information on Memory Optimization, p. 141*).

Structure of the CPU X13 0X memory (simplified representation):



Selecting Optimal EXEC File

Introduction

The simplest and most basic option is to download the optimal EXEC file for your requirements onto the PLC (see also *Installation Instructions*).

Depending on which EXEC file you select, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you install a 'combined EXEC file' and then only use one of the two language types in the user program, the program memory will not be used optimally.

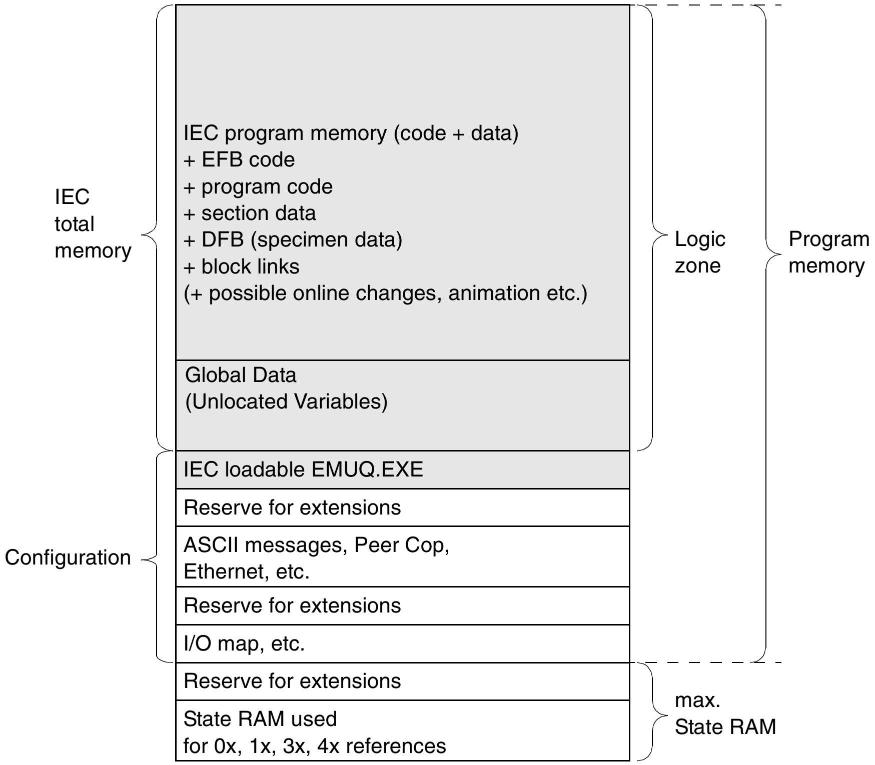
Therefore, decide which languages you want to use:

- *Exclusive Use of IEC, p. 151*
- *Exclusive Use of LL984, p. 152*
- *Joint Use of IEC and LL984, p. 153*

Exclusive Use of IEC

If you want to use IEC exclusively, download the EXEC file "QIEC_xxx.bin" (not available for CPU 424 02). Since this EXEC file does not contain an operating system, you have to download the IEC runtime system onto the PLC in the form of a loadable (EMUQ.exe) (related topics *Loadables, p. 96*). The loadable is downloaded into the program memory zone and takes up memory space.

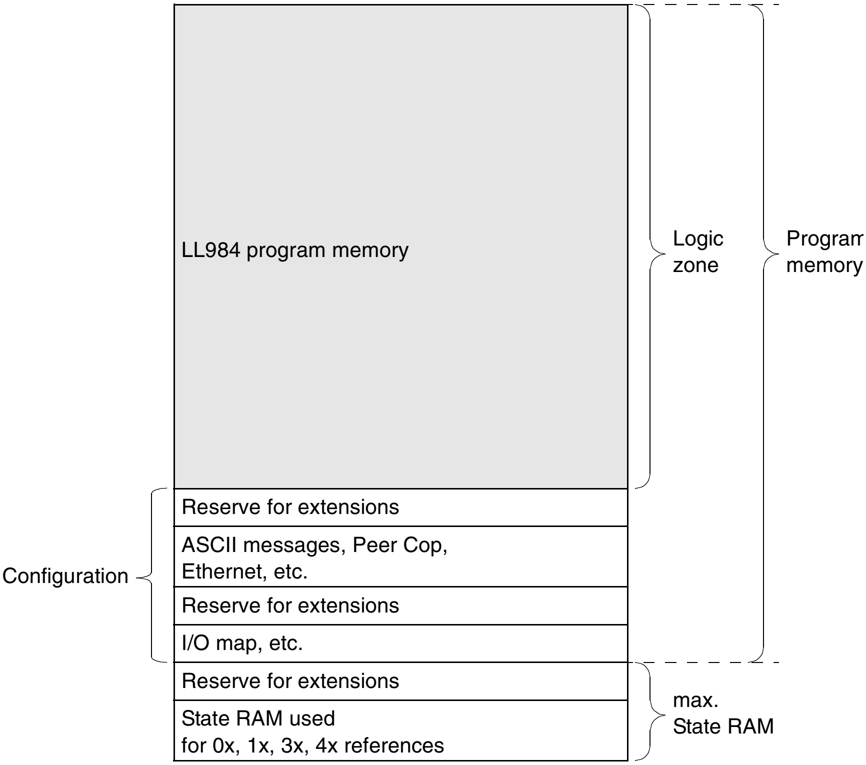
Structure of the CPU X13 0X memory with exclusive use of IEC:



Exclusive Use of LL984

If you want to use LL984 exclusively, download the EXEC file "Q186Vxxx.bin" for a CPU X13 0X and the EXEC file "Q486Vxxx.bin" for a CPU 424 02.

Structure of the CPU X13 0X memory with exclusive use of LL984:

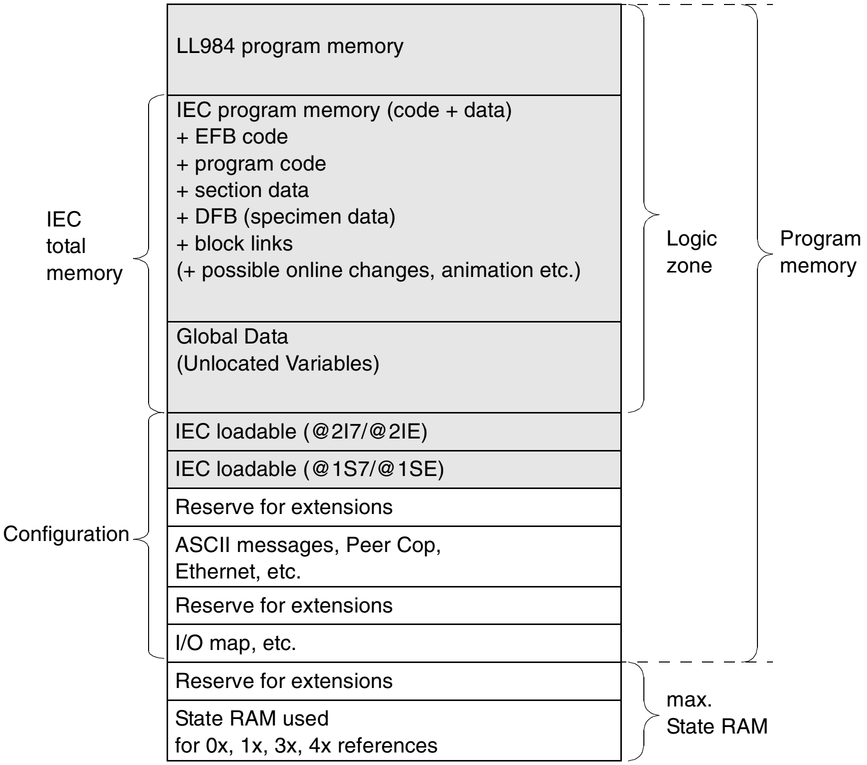


Joint Use of IEC and LL984

If joint use of IEC and LL984 is required, download the EXEC file "Q186Vxxx.bin" for a CPU X13 0X and the EXEC file zone "Q486Vxxx.bin" for a CPU 424 02. Since these EXEC files only contain the LL984 operating system, you have to download the IEC operating system onto the PLC in the form of loadables (@2I7/@2IE or @1S7/@1SE) (see also *Loadables, p. 96*). Both loadables will be downloaded into the program memory zone and occupy memory space.

> **Note:** Joint use of IEC and LL984 is not possible with the CPU 113 02 because its memory is too small for this application.

Structure of the CPU X13 0X memory with joint use of IEC and LL984:



LL984 program memory

IEC program memory (code + data)
+ EFB code
+ program code
+ section data
+ DFB (specimen data)
+ block links
(+ possible online changes, animation etc.)

Global Data
(Unlocated Variables)

IEC loadable (@2I7/@2IE)

IEC loadable (@1S7/@1SE)

Reserve for extensions

ASCII messages, Peer Cop,
Ethernet, etc.

Reserve for extensions

I/O map, etc.

Reserve for extensions

State RAM used
for 0x, 1x, 3x, 4x references

Using the Extended Memory (State RAM for 6x references)

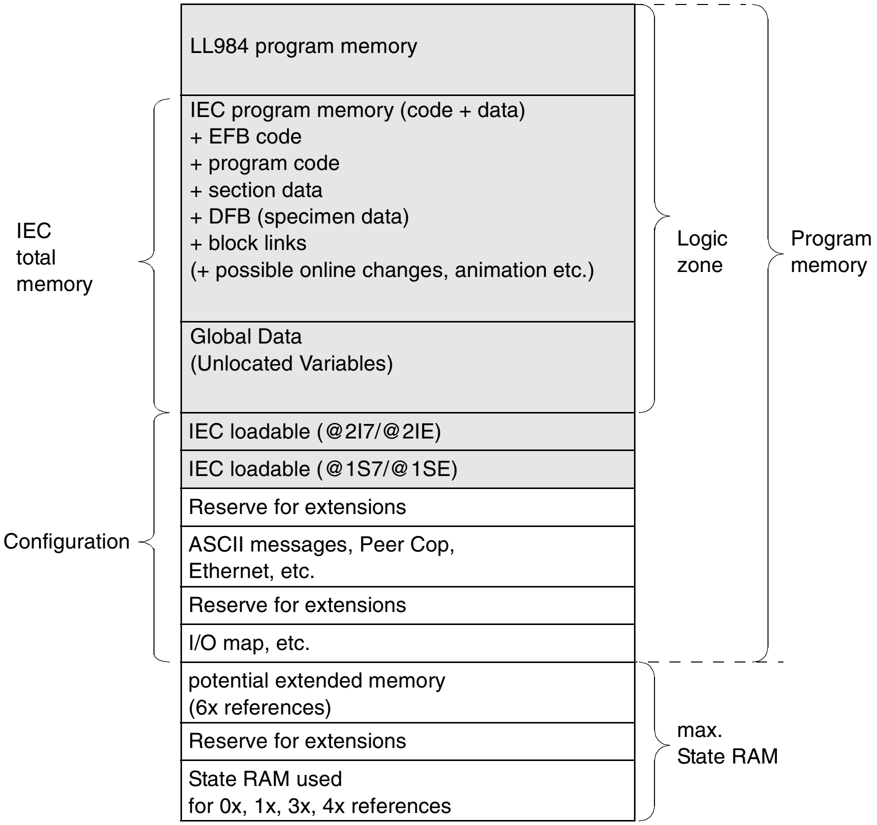
Introduction

If a CPU 213 04 or CPU 424 02 is used, you can make a zone in the state RAM available for the 6x references.

Note: 6x references are registers and can only be used with LL984 user programs.

Even if the state RAM memory zone is outside the program memory zone, the size of the state RAM influences the size of the program memory.

Using the extended memory (state RAM for 6x references):



If you do NOT use 6x

If you do not want to use any 6x references, you can, with a CPU 213 04, select whether to reserve state RAM 6x references or not.

Under **Project** → **PLC Configuration** → **PLC Selection** select from the **Memory Partition** the **48 K Logic / 32 K Memory**entry.

Note: With a CPU 424 02 there is no option for deactivating the 6x zone.

If you use 6x

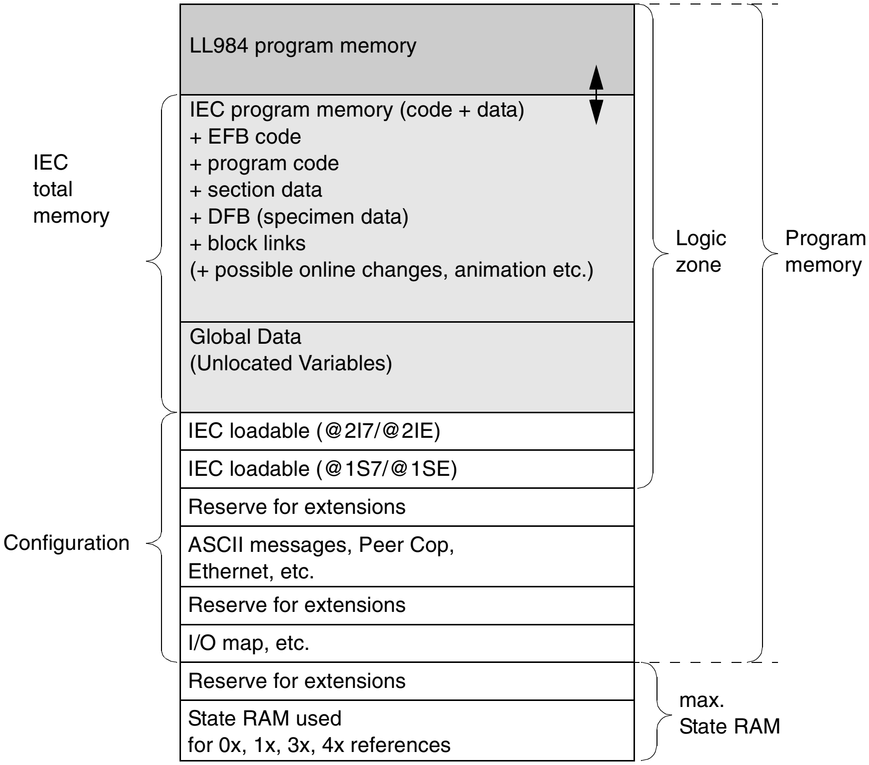
If you want to use 6x references, select under **Project** → **PLC Configuration** → **PLC selection** in the **Memory Partition** list box, the **32 K Logic / 64 K Memory**entry.

Harmonizing the IEC Zone and LL984 Zone

Introduction

With joint use of IEC and LL984 sections, the sizes of both zones should be harmonized with each other.

Harmonizing the IEC zone and LL984 zone:



Size of IEC Zone

The size of the total IEC memory and also the available space for LL984 data (user program) is determined by the memory size of the loadable @2I7 or @2IE.

You can define the memory size of the loadables in **Project** → **PLC Configuration** → **Loadables** → **Install @2I7 or @2IE** → **Edit...** → **Memory Size**.

The total size is given in paragraphs. A paragraph equals 16 bytes.

For the @1S7 or @1SE loadables, no memory size is needed. Ensure that "0" is specified here.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory, p. 158*.

Size of LL984 Zone

The size of the available memory for LL984 user programs is calculated using the following formula:

LL984 zone = available LL984 nodes – memory size of loadable @2I7/@2IE – size of loadables @2I7 or @2IE – size of loadables @1S7 or @1SE

When doing this calculation, it must be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are byte-oriented.

Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The loadable memory size is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory, p. 158*).

Harmonizing the Zones for Global Data and IEC Program Memory

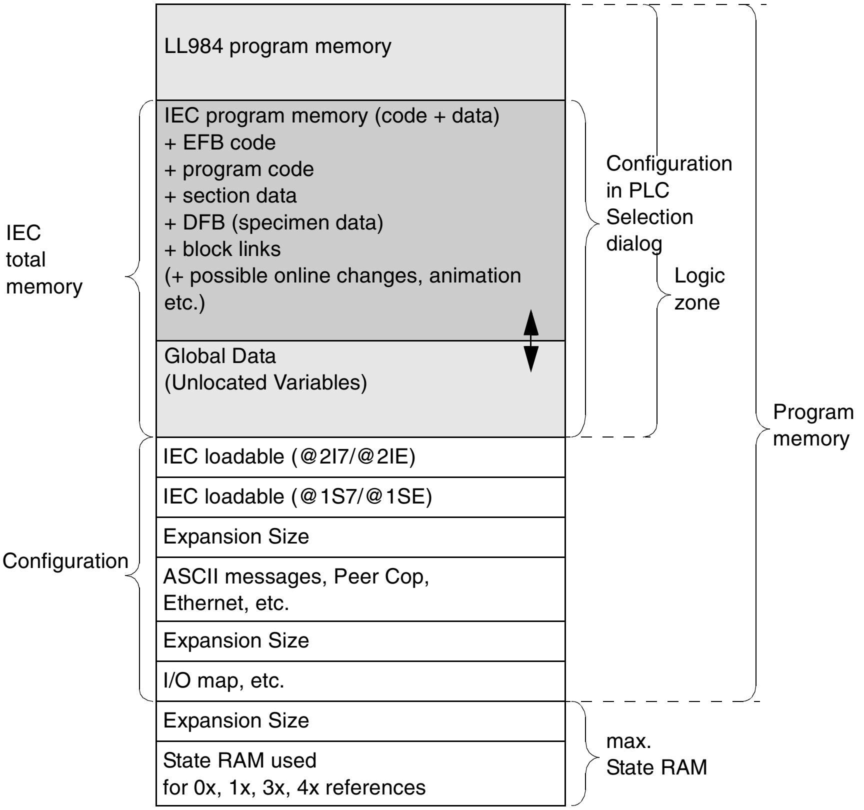
Introduction

The total IEC memory space, determined by the loadable memory size, (see Chapter *Harmonizing the IEC Zone and LL984 Zone, p. 156*) is made up of two zones:

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for IEC Program Memory and Global Data:



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** → **PLC Configuration** → **PLC selection** in the **IEC**zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** → **Memory statistics...** → **Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The loadable memory size is too small (see Chapter *Harmonizing the IEC Zone and LL984 Zone, p. 156*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.4 Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)

Introduction

Overview

This section describes the memory optimization for the Quantum CPUs 434 12(A) and 534 14(A/B).

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)	162
Harmonizing IEC Zone and LL984 Zone	164
Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))	169

General Information on Memory Optimization for Quantum CPU 434 12(A) and 534 14(A/B)

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project** → **PLC Configurator** in the configurations overview in the **PLC** zone. The memory size is given in nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

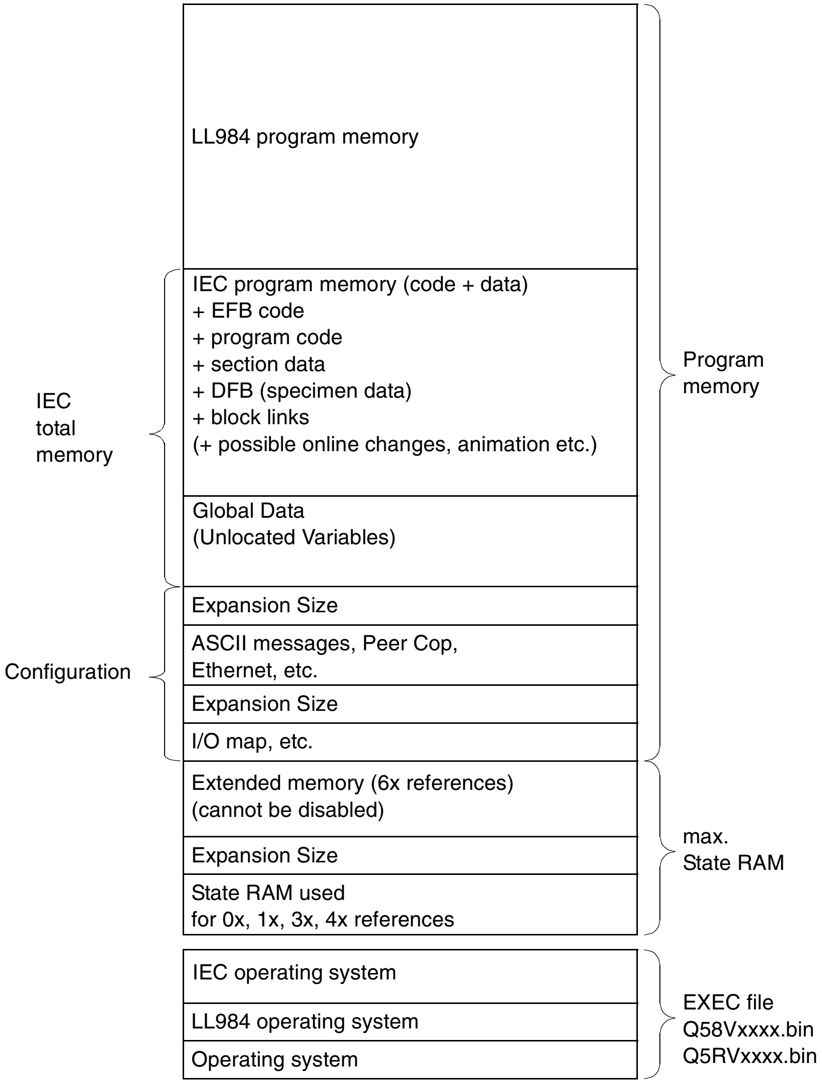
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Harmonizing IEC Zone and LL984 Zone, p. 164*
- *Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)), p. 169*

Note: Also note the PLC-independent possibilities for memory optimization (see *General Information on Memory Optimization, p. 141*).

Structure of the CPU 434 12(A) / 534 14(A/B) memory (simplified representation):



Harmonizing IEC Zone and LL984 Zone

Introduction

The EXEC file "Q58Vxxxx.bin" is required for the CPU 434 12 and 534 14.

The EXEC file "Q5RVxxxx.bin" is required for the CPU 434 12A and 534 14A/B (redesigned CPUs).

These EXEC files contain the runtime systems for IEC and LL984.

The sizes of the logic zones for IEC and LL984 should be harmonized with each other. The size of both zones can be defined in **Project** → **PLC Configurator** → **PLC selection**.

Depending on the size you select for the IEC zone, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you define a combined IEC and LL984 zone and then only use one of the two language types in the user program, the program memory will not be used optimally.

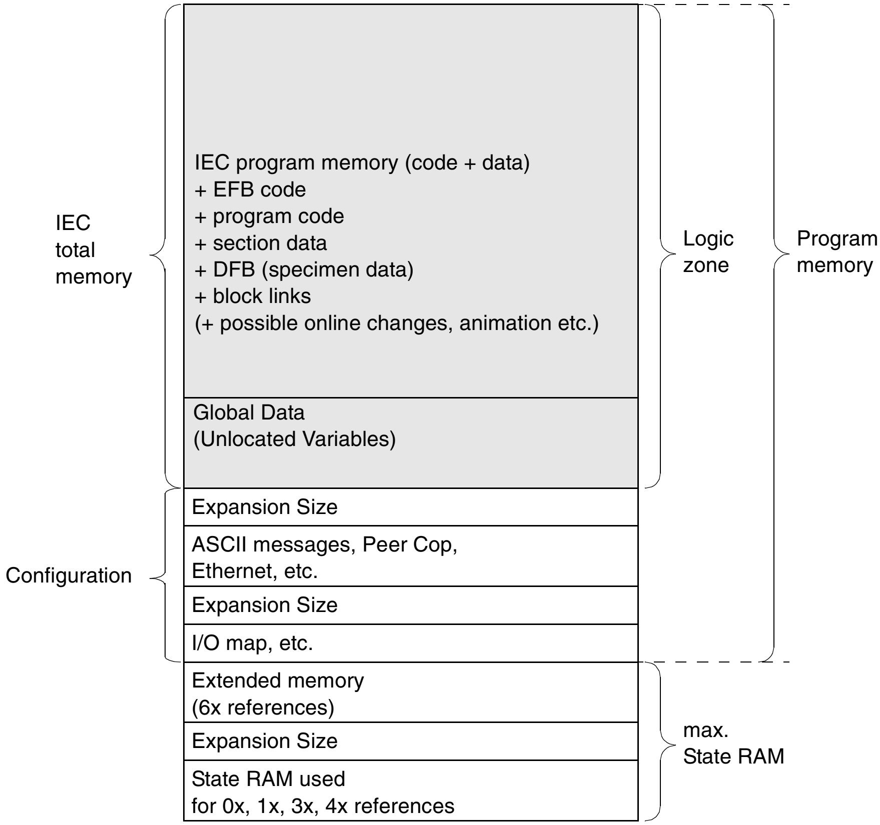
Therefore, decide which languages you want to use:

- *Exclusive Use of IEC, p. 165*
- *Exclusive Use of LL984, p. 166*
- *Joint Use of IEC and LL984, p. 167*

Exclusive Use of IEC

If you require exclusive use of the IEC, select in **Project** → **PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

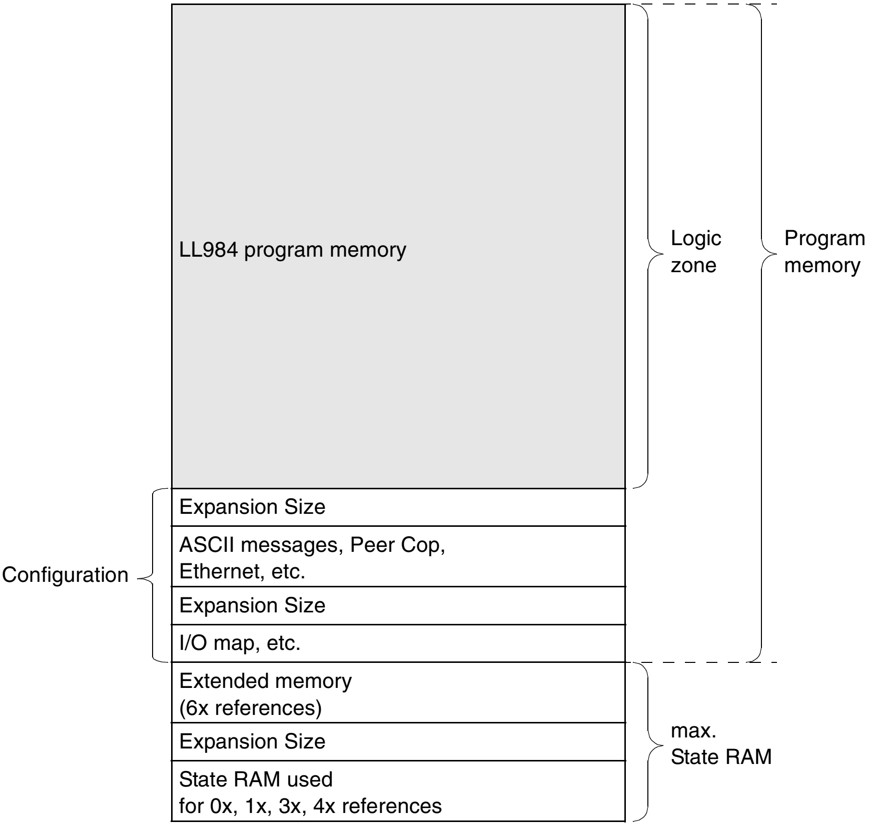
Structure of the CPU 434 12 (A)/ 534 14(A/B) memory with exclusive use of IEC:



Exclusive Use of LL984

If you require exclusive use of LL984, select from **Project** → **PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the **Disable** entry. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984 user program.

Structure of the CPU 434 12(A)/ 534 14(A/B) memory with exclusive use of LL984:



Joint Use of IEC and LL984

When using IEC and LL984 jointly, you should harmonize the sizes of both zones with each other.

By setting the **total IEC memory size** and **Global Data** you can automatically determine the size of the IEC program memory, and also the available space for LL984-data (user program).

The size of the available memory for LL984 user programs is calculated using the following formula:

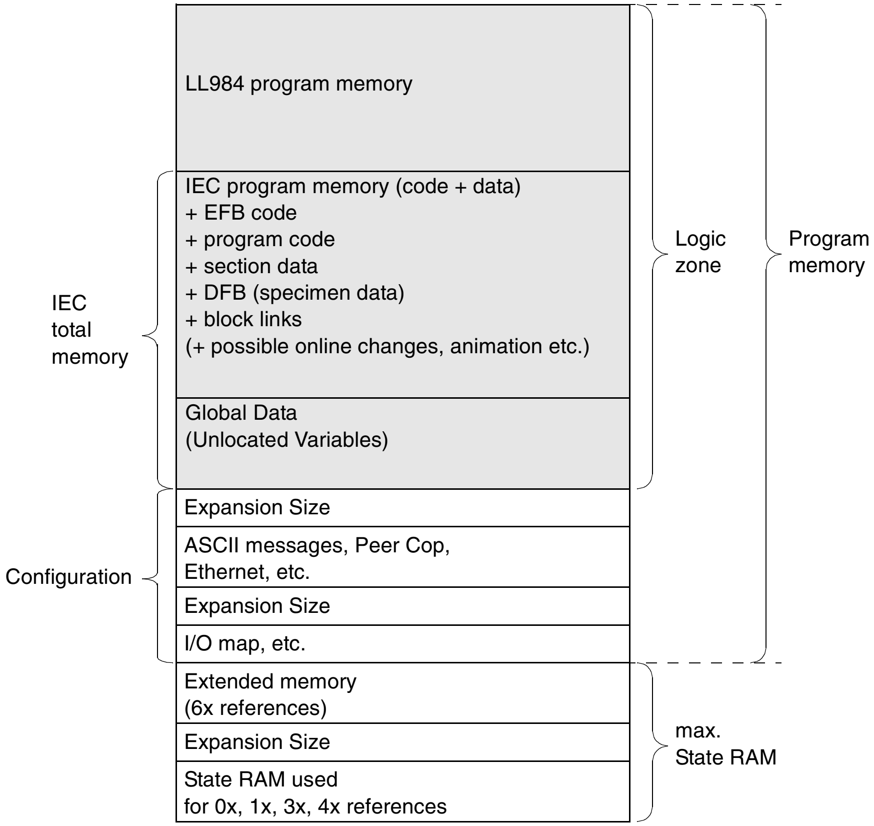
LL984 zone = available LL984 nodes - total IEC memory

When performing this calculation, it must however be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are kilobyte-oriented.

To set the total IEC memory, select from **Project** → **PLC Configuration** → **PLC selection** in the **IEC Operating System** list box, the **Enable** entry. The IEC zone is now enabled and you can enter the required memory size in the **Total IEC Memory** text box. The memory size is given in kilobytes.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory, p. 158*.

Structure of the CPU 434 12(A)/ 534 14(A/B) memory with exclusive use of IEC and LL984:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B)), p. 169*).

Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))

Introduction

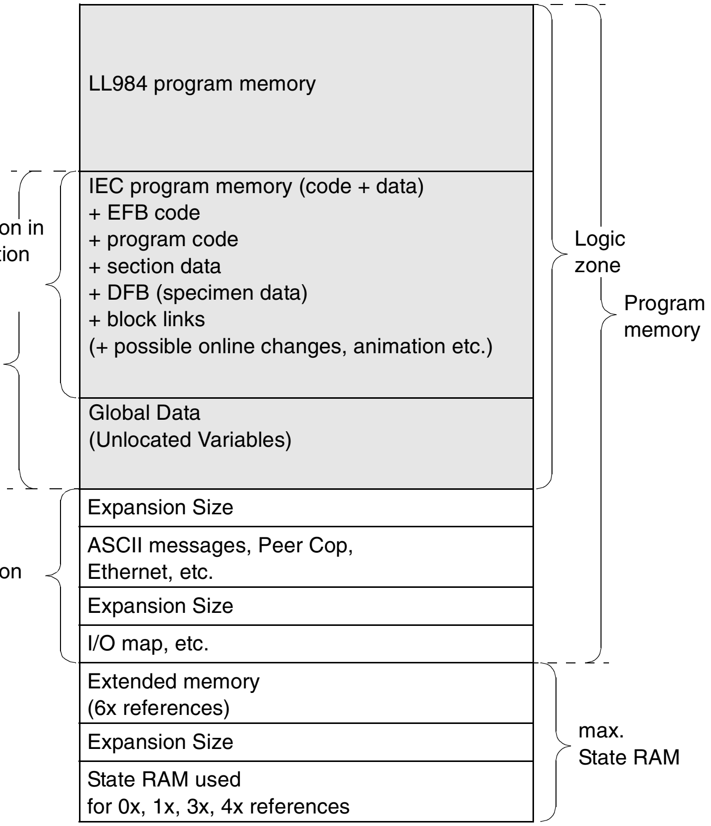
The fixed total IEC memory (see chapter *Harmonizing IEC Zone and LL984 Zone, p. 164*) is made up of two zones.

The total IEC memory space, determined by the loadable memory size, (see Chapter *Harmonizing the IEC Zone and LL984 Zone, p. 156*) is made up of two zones:

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (CPU 434 12(A) / 534 14 (A/B))



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** → **PLC Configuration** → **PLC selection** in the **IEC**zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** → **Memory statistics...** → **Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Harmonizing IEC Zone and LL984 Zone, p. 164*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.5 Memory optimization for Compact CPUs

At a Glance

Overview

This Section describes the memory optimization for Compact CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Compact CPUs	174
Harmonizing IEC Zone and LL984 Zone	176
Harmonizing the Zones for Global Data and IEC Program Memory (Compact)	181

General Information on Memory Optimization for Compact CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project** → **PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

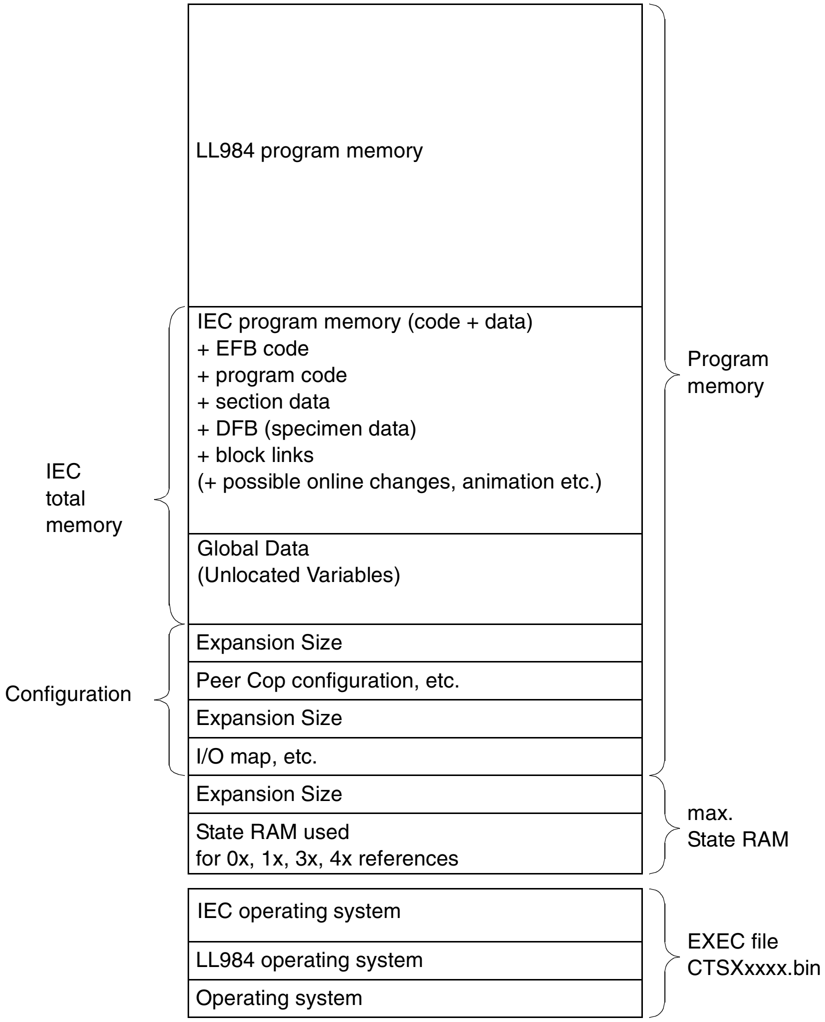
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Harmonizing IEC Zone and LL984 Zone, p. 176*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), p. 181*

Note: Also note the PLC-independent possibilities for memory optimization (see *General Information on Memory Optimization, p. 141*).

Structure of a Compact CPU memory (simplified representation)



Harmonizing IEC Zone and LL984 Zone

Introduction

The IEC zone "CTSXxxxx.bin", required for Compact CPUs, contains the runtime systems for IEC and LL984 (see also *Installation instructions*).

The sizes of the logic zones for IEC and LL984 should be harmonized with each other. You can define the size of both zones in **Project** → **PLC Configurator** → **PLC Selection**.

Depending on the size you select for the IEC zone, zones will be reserved in the program memory of the PLC for IEC and/or LL984 programs. Therefore, if you define a combined IEC and LL984 zone and then only use one of the two language types in the user program, the program memory will not be used optimally.

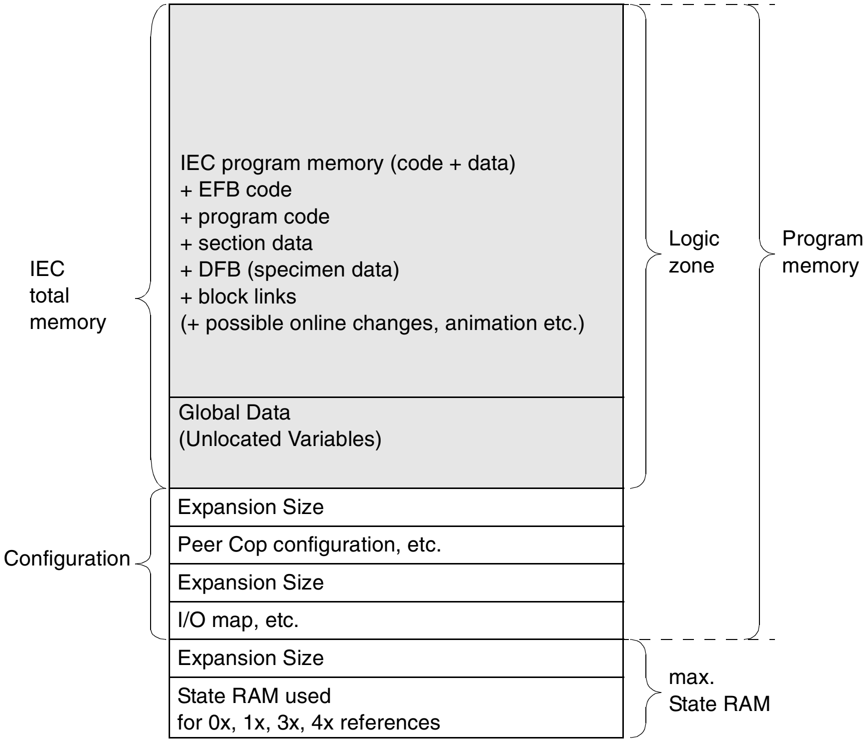
Therefore, decide which languages you want to use:

- *Exclusive Use of IEC, p. 177*
- *Exclusive Use of LL984, p. 178*
- *Joint Use of IEC and LL984, p. 179*

Exclusive Use of IEC

If you require exclusive use of the IEC, select in **Project** → **PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

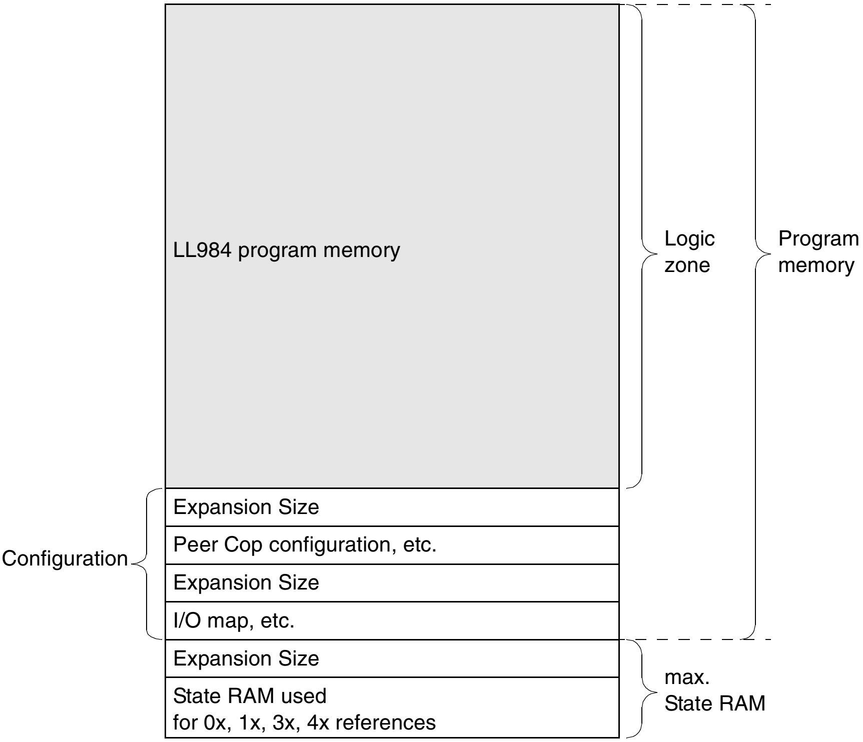
Structure of the Compact CPU memory with exclusive use of IEC



Exclusive Use of LL984

If you require exclusive use of LL984, select from **Project** → **PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the **Disable** entry. This will completely switch off the IEC zone and the entire logic zone will be made available for the LL984 user program.

Structure of the Compact CPU memory with exclusive use of LL984



Joint Use of IEC and LL984

When using IEC and LL984 jointly, you should harmonize the sizes of both zones with each other.

By setting the **total IEC memory size** and **Global Data** you can automatically determine the size of the IEC program memory, and also the available space for LL984-data (user program).

The size of the available memory for LL984 user programs is calculated using the following formula:

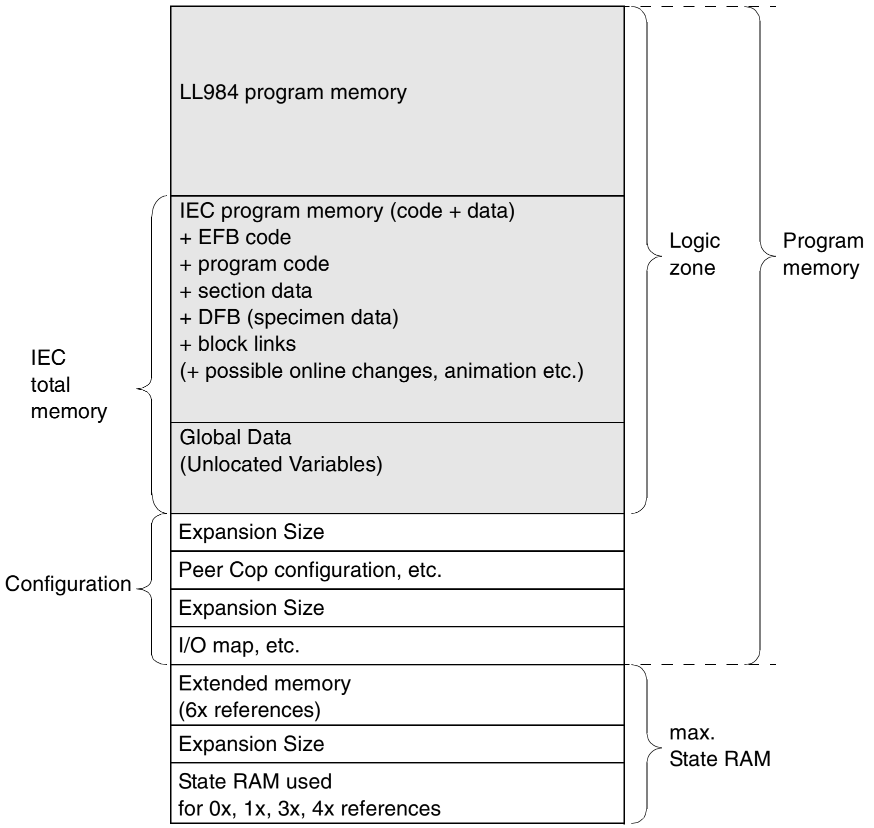
LL984 zone = available LL984 nodes - total IEC memory

When performing this calculation, it must however be ensured that the size of the LL984 zone is node-oriented and the remaining instructions are kilobyte-oriented.

To set the total IEC memory, select from **Project** → **PLC Configuration** → **PLC selection** in the **IEC Operating System** list box, the **Enable** entry. The IEC zone is now enabled and you can enter the required memory size in the **Total IEC Memory** text box. The memory size is given in kilobytes.

The fixed total IEC memory size is again made up of several zones. You will find the explanation of how to harmonize these zones vertically in the chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), p. 181*.

Structure of the Compact Memory with joint use of IEC and LL984:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Compact), p. 181*).

Harmonizing the Zones for Global Data and IEC Program Memory (Compact)

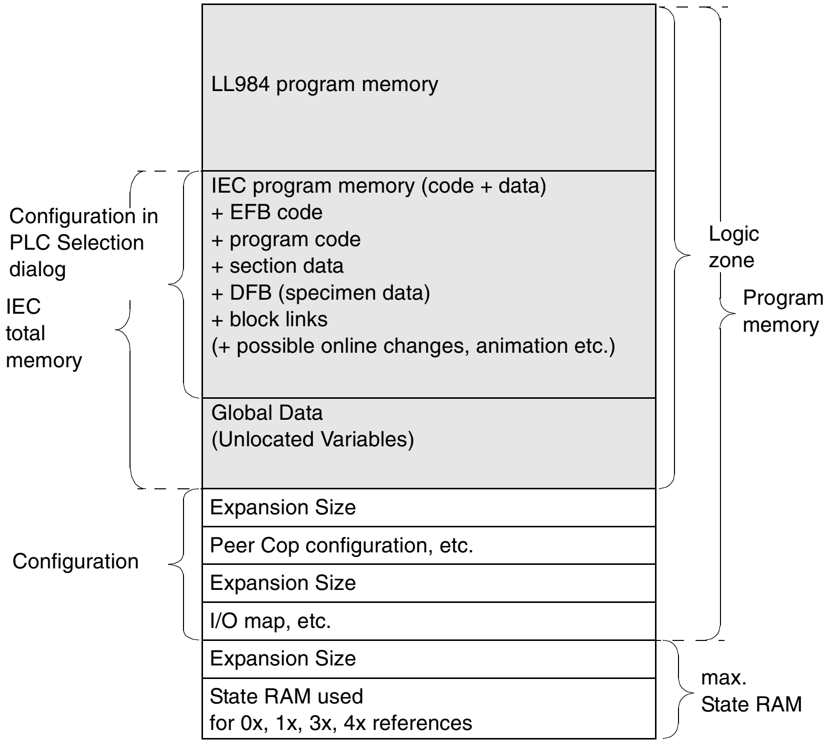
Introduction

The fixed total IEC memory (see chapter *Harmonizing IEC Zone and LL984 Zone, p. 176*) is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (Compact):



Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** → **PLC Configuration** → **PLC selection** in the **IEC** zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** → **Memory statistics...** → **Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Harmonizing IEC Zone and LL984 Zone, p. 176*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.6 Memory optimization for Momentum CPUs

Introduction

Overview

This Section describes the memory optimization for Momentum CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Momentum CPUs	184
Selecting Optimal EXEC file	186
Harmonizing the Zones for Global Data and IEC Program Memory (Momentum)	187

General Information on Memory Optimization for Momentum CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project** → **PLC Configuration** in the configurations overview in the **PLC** zone. The entry for the memory size is given in Nodes for LL984 (1 node equals 11 bytes) and in kilobytes for IEC.

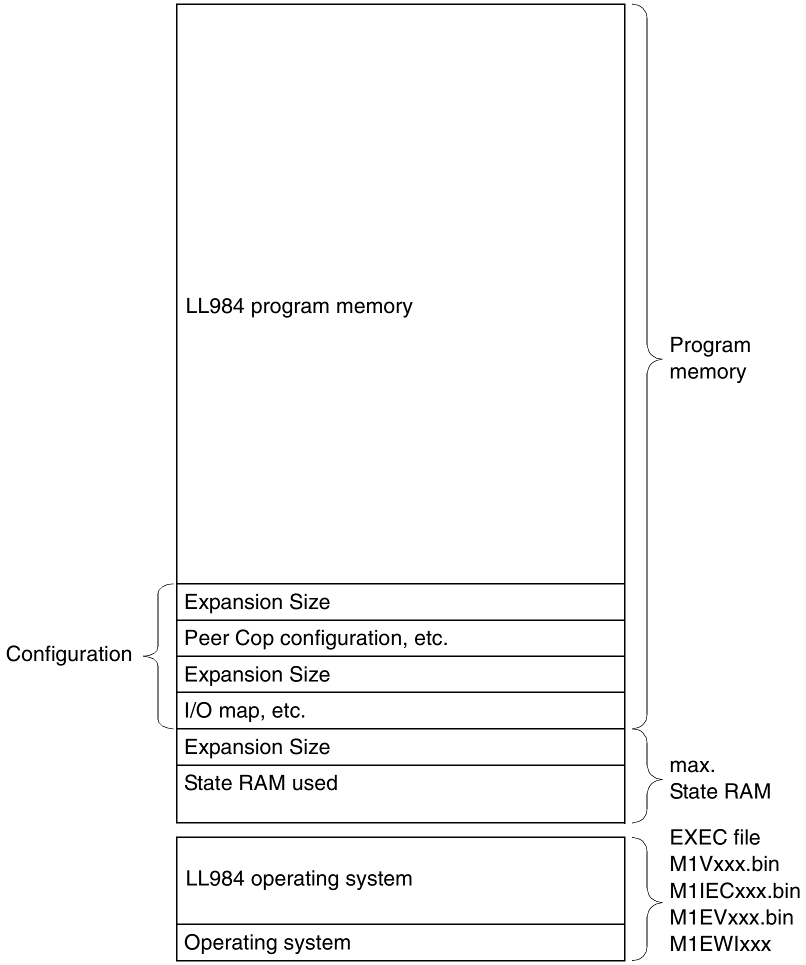
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Selecting Optimal EXEC file, p. 186*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Momentum), p. 187*

Note: Also note the PLC-independent possibilities for memory optimization (see *General Information on Memory Optimization, p. 141*).

Structure of a Momentum CPU memory (simplified representation):



Selecting Optimal EXEC file

Introduction

It is not possible to use IEC and LL984 jointly in Momentum.

Using IEC

EXEC file assignment during IEC use:

171 CBB	M1IVxxxE	MPSV100e.BIN
970 30	-	x

171 CCS	M1IVxxxE	M1EVxxxE
760 00	x	-
760 10	x	-
780 10	x	-
960 30	-	x
980 30	-	x

Using LL984

EXEC file assignment during LL984 use:

171 CBB	M1LLVxxx	M1MVxxxE
970 30	x	-

171 CCS	M1LLVxxx	M1EVxxx
700 10	x	-
700/780 00	x	-
760 00	x	-
760 10	x	-
780 10	x	-
960 20	-	x
960 30	-	x
980 20	-	x
980 30	-	x

Harmonizing the Zones for Global Data and IEC Program Memory (Momentum)

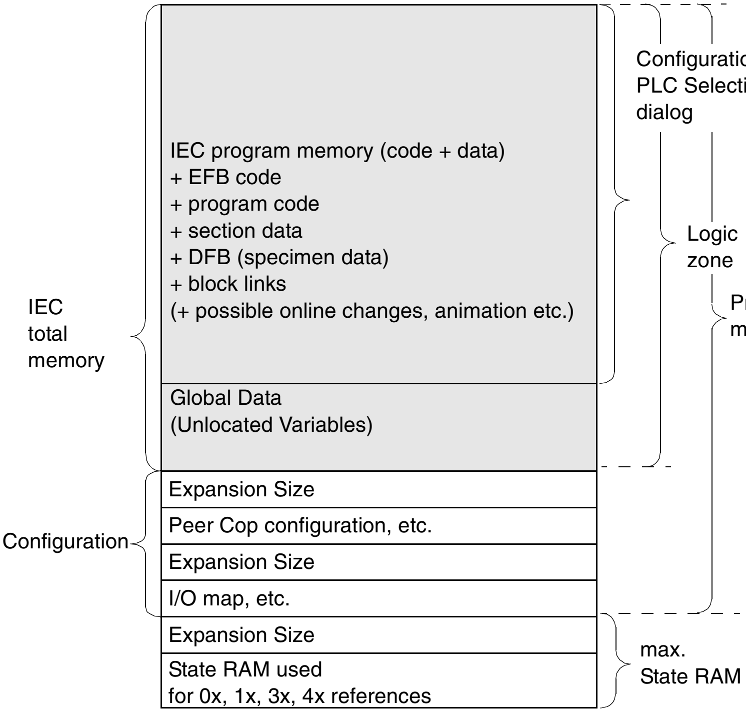
Introduction

The logic zone for the total IEC memory is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global data and IEC Program Memory (Momentum 171 CCS 760 00-IEC):



Size of the IEC Program Memory Zone

The settings for the IEC user program zone are available in **Online** → **Memory statistics...** → **Memory statistics** in the **Configured** text box. This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, because hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables and block links) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** → **Memory statistics...** → **Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are two possible reasons for an error message, saying that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

6.7 Memory optimization for Atrium CPUs

At a Glance

Overview

This Section describes the memory optimization for Atrium CPUs.

What's in this Section?

This section contains the following topics:

Topic	Page
General Information on Memory Optimization for Atrium CPUs	190
Use of IEC	191
Harmonizing the Zones for Global Data and IEC Program Memory (Atrium)	193

General Information on Memory Optimization for Atrium CPUs

Logic Memory

The program memory zone, in which the user program is located, is called the logic zone. This zone therefore determines the maximum size of your user program.

The current size of the logic zone is displayed under **Project** → **PLC Configurator** in the configurations overview in the **PLC** zone. The memory size is given in kilobytes for IEC.

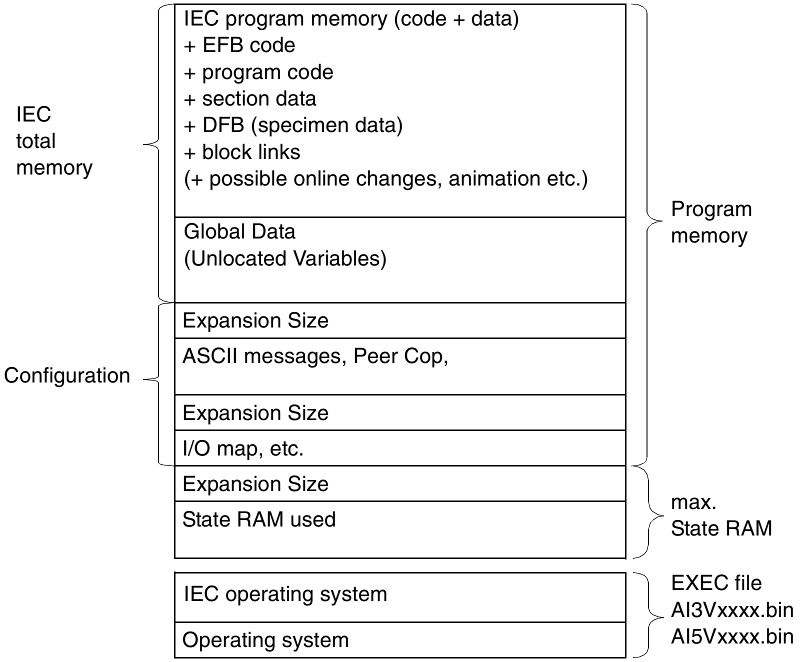
Optimizing the Logic Memory

You have various possibilities for optimising the logic memory to suit your requirements:

- *Use of IEC, p. 191*
- *Harmonizing the Zones for Global Data and IEC Program Memory (Atrium), p. 193*

Note: Also note the PLC-independent possibilities for memory optimization (see *General Information on Memory Optimization, p. 141*).

Structure of the Atrium CPU Memory (simplified representation):



Use of IEC

Introduction

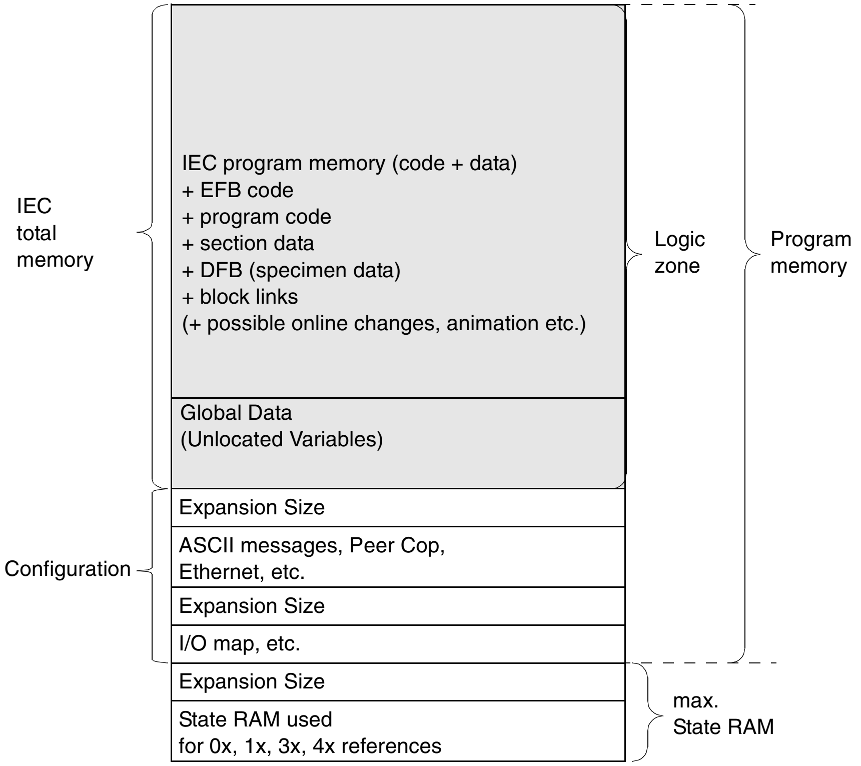
The EXEC files required for the CPUs of the Atrium family contain the operating systems for IEC (see also *Installation Instructions*).

When using the Atrium 180 CCO 121 01, load the EXEC file "AI3Vxxxx.bin".

When using the Atrium 180 CCO 241 01and 180 CCO 241 11 load the EXEC file "AI5Vxxxx.bin".

Select in **Project** → **PLC Configuration** → **PLC Selection** in the **IEC Operating System** list box, the entry **Enable** and drag the **total IEC memory** slider to the right hand margin (highest value). This will completely switch off the LL984 zone and the entire logic zone will be made available for the IEC user program.

Structure of the Atrium CPU memory with exclusive use of IEC:



Error Message during Download of Program

There are three possible causes for an error message, which says that the user program is too large for the PLC memory, appearing during download:

1. The memory is currently too small.
2. The logic zone is too small (see current chapter).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see chapter *Harmonizing the Zones for Global Data and IEC Program Memory (Atrium), p. 193*).

Harmonizing the Zones for Global Data and IEC Program Memory (Atrium)

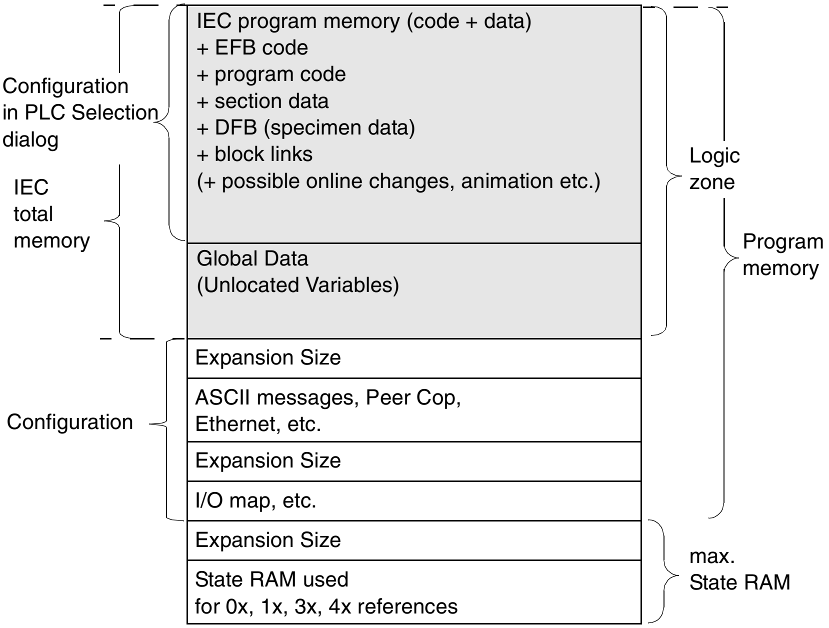
Introduction

The fixed total IEC memory (see chapter *Use of IEC, p. 191*) is made up of two zones.

- **IEC Program Memory**
  - comprising the EFB codes,
  - the program codes,
  - the section data,
  - the DFB specimen data,
  - the block links,
  - possibly data from online changes,
  - possibly animation data etc.
- **Global Data**
  - comprising the Unlocated Variables

The zones for global data and IEC program memory can be harmonized with one another.

Harmonizing the Zones for Global Data and IEC Program Memory (Atrium):



IEC program memory (code + data)
+ EFB code
+ program code
+ section data
+ DFB (specimen data)
+ block links
(+ possible online changes, animation etc.)

Global Data
(Unlocated Variables)

Expansion Size

ASCII messages, Peer Cop,
Ethernet, etc.

Expansion Size

I/O map, etc.

Expansion Size

State RAM used
for 0x, 1x, 3x, 4x references

Size of the IEC Program Memory Zone

You change the settings for the IEC program memory in **Project** → **PLC Configuration** → **PLC selection** in the **IEC** zone. Enter the size of the total IEC memory and the global data, so that the IEC program memory size will be calculated (IEC program memory size = total IEC memory - global data). This setting is only possible when the PC and PLC are offline. If you do not use any or only a few unlocated variables and have no or only a few block links, you can select the IEC program memory as very large, since hardly any memory is needed for global data.

Size of the Zone for Global Data

The zone for global data (unlocated variables) is calculated using the following formula:

Zone for global data = memory size of the loadable - IEC program memory

The current content of the individual zones (EFBs, specimen data, user program etc.) is displayed under **Online** → **Memory statistics...** → **Memory statistics**. This display is only possible when the PC and PLC are online.

Error Message during Download of Program

There are three possible reasons for an error message, which says that the user program is too large for the PLC memory, appearing while downloading the program onto the PLC:

1. The memory is currently too small.
2. The total IEC memory size is too small (see Chapter *Use of IEC, p. 191*).
3. The zone for global data and the IEC program memory zone are not optimally harmonized (see current chapter).

Function Block language FBD

7

At a Glance

Overview

This Chapter describes the Function Block language FBD which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	General information about FBD Function Block	197
7.2	FBD Function Block objects	199
7.3	Working with the FBD Function Block langauge	209
7.4	Code generation with the FBD Function Block language	215
7.5	Online functions of the FBD Function Block language	217
7.6	Creating a program with the FBD Function Block language	219

7.1 General information about FBD Function Block

General information on Function Block language FBD

At a Glance

The objects of the programming language FBD (Function Block Diagram) help to divide a section into a number of:

- EFBs (Elementary Functions and Elementary Function Blocks) (see *EFB, p. 200*),
- DFBs (Derived Function Blocks) (see *DFB, p. 202*) and
- UDEFBs (User-defined Functions and Function Blocks) (see *UDEFB, p. 203*).

These objects, combined under the name FFBs, can be linked with each other by:

- Links (see *Link, p. 204*) or
- Current parameters (see *Actual parameters, p. 205*).

Expansive logic can also be placed in the FBD section in the form of macros (see also *Macros, p. 511*).

Theoretically, each section can contain as many FFBs and also as many inputs and outputs as required. However, it is advisable to subdivide a whole program in logic units, that is to say in different sections.

Comments can be provided for the logic of the section with text objects (see *Text Object, p. 207*).

Processing sequence

The processing sequence of the individual FFBs in an FBD section is determined by the data flow within the section (see also *FFB Execution Order, p. 212*).

Editing with the keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (see also *Short Cut Keys in the FBD and SFC Editor, p. 843*)

IEC conformity

For a description of the IEC conformity of the FBD programming language see *IEC conformity, p. 857*.

7.2 FBD Function Block objects

At a Glance

Overview

This section describes the FBD Function Block objects.

What's in this Section?

This section contains the following topics:

Topic	Page
Functions and Function Blocks (FFBs)	200
Link	204
Actual parameters	205
Text Object	207

Functions and Function Blocks (FFBs)

Introduction

FFB is the generic term for:

- EFB (Elementary Function and Elementary Function Block) (see *EFB, p. 200*)
- DFB (Derived Function Block) (see *DFB, p. 202*)
- UDEFB (Derived Elementary Function and Derived Elementary Function Block) (see *UDEFB, p. 203*)

EFB

EFB is the generic term for:

- Elementary Function (see *Elementary Function, p. 200*)
- Elementary Function Block (see *Elementary Function Block, p. 201*)

EFBs are functions and function blocks that are available in Concept in the form of libraries. The logic of EFBs is built in C programming language and cannot be changed in the FBD editor.

Elementary Function

Functions have no internal conditions. If the input values are the same, the value at the output is the same for all executions of the function. E.g. the addition of two values gives the same result at every execution.

An Elementary Function is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function, that is the function type, is displayed in the center of the frame. The function counter is displayed above the frame.

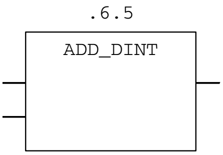
The function counter cannot be changed and always has an .n.m. structure.

.n = current section number

.m = current function number

Functions are only executed in FBD if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, p. 203*).

Elementary Function



Elementary Function Block

Function blocks have internal conditions. If the inputs have the same values, the value at the output at every execution is another value. E.g. with a counter, the value on the output is incremented.

A function block is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function block, that is the function block type, is displayed in the center of the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

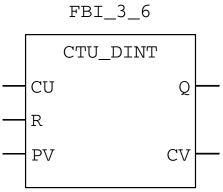
m = Number of the FFB object in the section (current number)

The instance name can be edited in the **Object** → **Properties** dialog box of the function block. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must correspond to the IEC name conventions, otherwise an error message occurs.

> **Note:** In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the menu command **Options** → **Preferences** → **IEC Extensions...** → **Permit Leading Figures in Identifiers** will enable this.

Function blocks are only executed in FBD if the input EN=1 or if the input EN is grayed out (related topics *EN and ENO, p. 203*).

Elementary Function Block



DFB

Derived Function Blocks (DFBs) are function blocks that have been defined in Concept DFB.

With DFBs, there is no distinction between functions and function blocks. They are always treated as function blocks regardless of their internal structure.

A DFB is represented graphically as a frame with double vertical lines and with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The DFB name is displayed centrally within the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

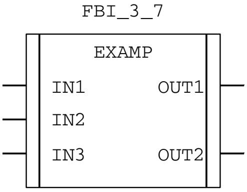
m = Number of the FFB object in the section (current number)

The instance name can be edited in the **Object** → **Properties** dialog box of the DFB. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must correspond to the IEC name conventions, otherwise an error message occurs.

Note: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the menu command **Options** → **Preferences** → **IEC Extensions...** → **Permit Leading Figures in Identifiers** will enable this.

Derived function blocks are only executed in FBD if the input EN=1 or if the input EN is grayed out (related topics *EN and ENO, p. 203*).

Derived Function Block



UDEFB

UDEFB is the generic term for:
- User-defined Elementary Function
- User-defined Elementary Function Block

UDEFBs are functions and function blocks that have been programmed with Concept EFB in C++ programming language and are available in Concept in the form of libraries.

In Concept, there is no functional difference between UDEFBs and EFBs.

EN and ENO

With all FFBs, an EN input and an ENO output can be configured.

The configuration of EN and ENO is switched on or off in the **FFB Properties** dialog box. The dialog box can be called up with the **Objects** → **Properties...** menu command or by double-clicking on the FFB.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is automatically set to "0" in this case.

If the value of EN is equal to "1", when the FFB is called up, the algorithms which are defined by the FFD will be executed. After successful execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during execution of these algorithms, ENO will be set to "0".

The output behavior of the FFBs in FBD does not depend on whether the FFBs are called up without EN/ENO or with EN=1.

Link

Description

Links are connections between FFBs.

Several links can be connected with one FFB output. The link points are identified by a filled-in circle.

Data Types

The data types of the inputs/outputs to be linked must be the same.

Creating Links

Links can be created using **Objects** → **Link**.

Editing Links

Links can be edited in select mode. An overlap with other objects is permitted.

Configuring Loops

No loop can be configured with links because in this case, the execution order in the section cannot be determined uniquely. Loops must be resolved with actual parameters (see *Configuring Loops, p. 214*).

Actual parameters

At a Glance

In the program runtime, the values from the process or from other actual parameters are transferred to the FFB over the actual parameters and then re-emitted after processing.

These actual parameters can be:
- direct addresses (see *Direct addresses, p. 46*)
- Located variables (see *Variables, p. 43*)
- Unlocated variable (see *Variables, p. 43*)
- Constants (see *Constant variables, p. 44*)
- Literals (see *Literals (values), p. 45*)

Direct addresses

The information on/display of direct addresses can be given in various formats. The display format is set in the dialog box **Options** → **Presettings** → **Joint**. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:
- **Standard format (400001)**
  The five-character address comes directly after the first digit (the Reference).
- **Separator format (4:00001)**
  The first digit (the Reference) is separated from the following five-character address by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the following address by a colon (:), and the leading zeros of the address are not given.
- **IEC format (QW1)**
  In first place, there is an IEC identifier, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

Data types

The data type of the actual parameter must match the data type of the input/output. The only exceptions are generic inputs/outputs, of which the data type is determined by the formal parameter. If all actual parameters consist of literals, a suitable data type is selected for the Function Block.

Initial values

FFBs, which use actual parameters on the inputs that have not yet received any value assignment, work with the initial values of these actual parameters.

Unconnected inputs

Note: Unconnected FFB inputs are specified as "0" by default.

Text Object

At a Glance

Text can be positioned in the form of text objects using FBD Function Block language. The size of these text objects depends on the length of the text. The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units. Text objects may not overlap with FFBs; however they can overlap with links.

Memory space

Text objects occupy no memory space on the PLC because the text is not downloaded onto the PLC.

7.3 Working with the FBD Function Block langauge

At a Glance

Overview

This section describes working with the FBD Function Block object language..

What's in this Section?

This section contains the following topics:

Topic	Page
Positioning Functions and Function Blocks	210
FFB Execution Order	212
Configuring Loops	214

Positioning Functions and Function Blocks

Selecting FFBs

Using **Objects** → **Select FFB...** you can open a dialog for selecting FFBs. This dialog is modeless, that is, it is not automatically closed once an FFB is positioned, but remains open until you close it. If you have several FBD sections open, and invoke the dialog, only one dialog box is opened that is available for all sections. The dialog box is not available for any other sections (non-FBD editor). If the FBD sections are changed into icons (minimize window), the dialog box is closed. If one of the FBD section icons is called up again, the dialog box is automatically re-opened.

The first time Concept is started the FFB is displayed oriented to the library. This means that, when selecting an FFB, the **Library** command button must first of all be used to select the corresponding library. Then you can select the corresponding **Group** in the list box. Now, you can select the required FFB from the EFB type list.

If you do not know which library/group the FFB required is in, you can invoke an FFB-oriented dialog with the **Sorted by FFB** command button. This contains all FFBs of all libraries and groups in an alphabetical list.

After each subsequent project start, the view you selected appears.

Once the FFB has been selected, its position in the section must be selected. The cursor becomes a small FFB and the cross shows the position (upper left corner of the FFB) where the FFB is positioned. The FFB is positioned by clicking on the left-hand mouse button.

Positioning FFBs (Functions and Function Blocks)

In the FBD function block language editor, the window appears with a logic grid. FFBs (see *Functions and Function Blocks (FFBs), p. 200*) are aligned in this grid as they are positioned. If FFBs are positioned outside of the section frame or if there is overlapping with another FFB, an error warning will appear and the FFB will not be positioned. Actual parameters may overlap another object when being positioned at an FFB input/output, but they must not go outside the limits of the section frame. If a link to another FFB is established, this link is checked. If this link is not permitted, a message is received, and the link is not established. When links are created, overlaps and crossing with other links and FFBs are permitted. If an FFB is selected, the comment relating to it is displayed in the first column of the status line. If an actual parameter is selected, its name and, if applicable, its direct address, its I/O map and its comment are displayed in the first column of the status line.

Change FFB Type

With the **Objects** → **Replace FFBs...** menu command the FFBs already positioned in the section can be replaced with FFBs of another type (e.g. an AND with an OR). The variables given to the FFB remain if the data type and position of the inputs/ outputs are the same as the "old" and the new FFB.

Note: FFBs with inputs / outputs of the ANY data type (generic FFBs) cannot be replaced.

FFB Execution Order

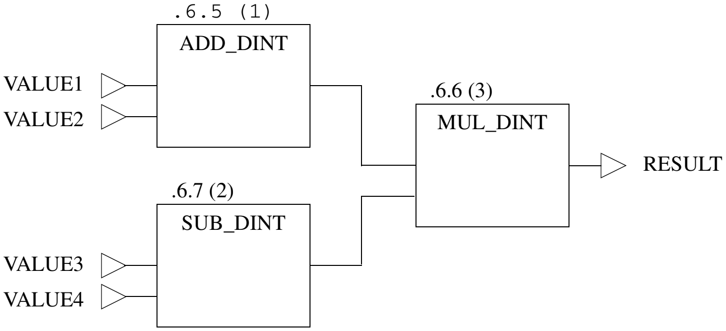
Introduction

The execution order is first determined by the order when positioning the FFB. If the FFBs are then linked graphically, the execution order is determined by the data flow.

Display FFB Execution Order.

The execution order can also be displayed with the **Objects** → **FFB Execution Order** menu command. This is represented by the execution number (number in brackets behind the instance name or function counter).

Show execution order of the FFBs



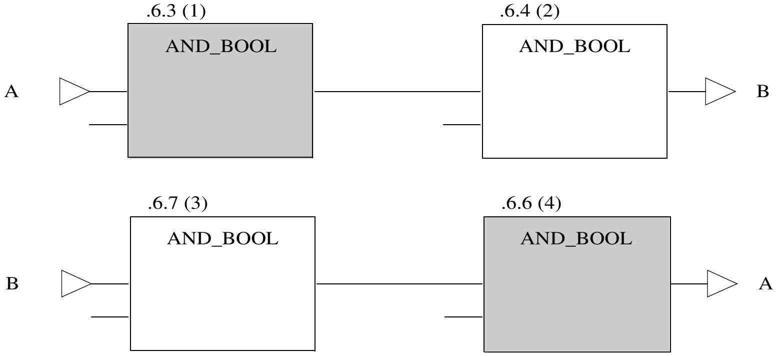
Change FFB Execution Order

The execution order can be specifically changed afterwards with the menu command **Objects** → **Change FFB Execution Order**, but only if the rules regarding data flow are not broken.

Changing the execution order of two networks which are in one loop

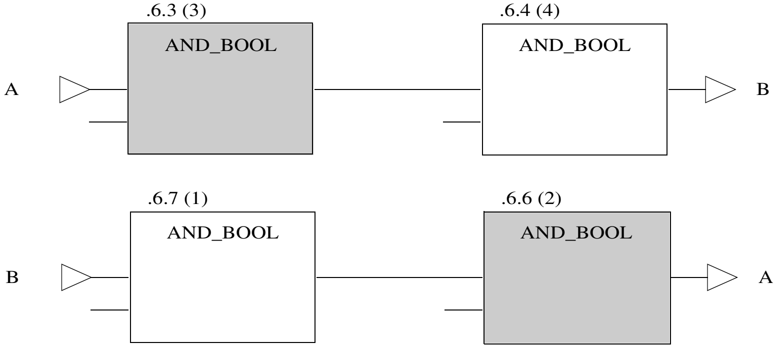
This change can only be made when the two FFBs are linked by the feedback variable of the loop.

Step 1: Select the two FFBs.



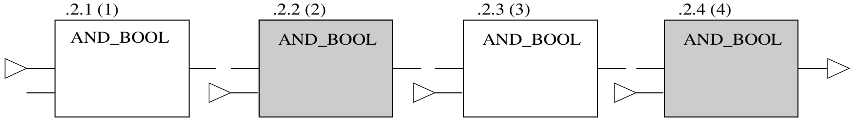
Step 2: Press the menu command **Change FFB-execution sequence**.

Result: The execution sequence has changed as follows:

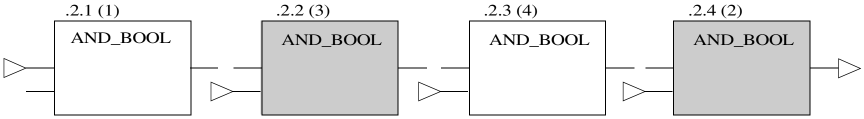


Changing the execution order of FFBs which are executed according to the positioning order

The change operation permits the creation of a different, desired order (sometimes step by step if several FFBs are involved).



Result: The execution sequence has changed as follows:

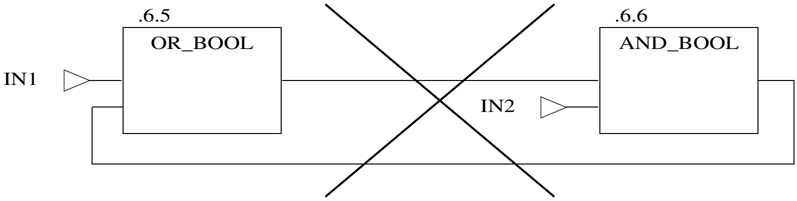


Configuring Loops

Non-permitted Loops

Configuring loops exclusively via links is not permitted, as it is not possible to uniquely set the data flow (the output of one FFB is the input of the next FFB, and the output of this one is the input of the first).

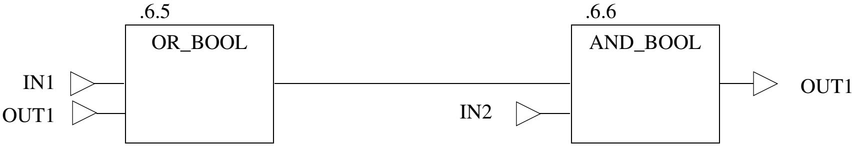
Non-permitted Loops via Links



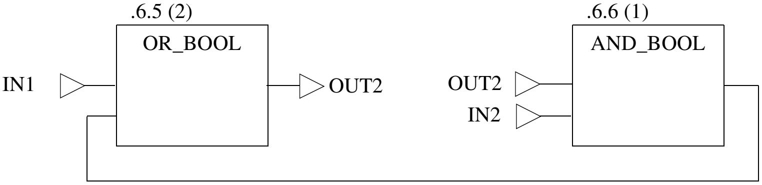
Resolution using an Actual Parameter

This type of logic must be resolved using actual parameters so that the data flow can be determined uniquely.

Resolved loop using an actual parameter: Variant 1



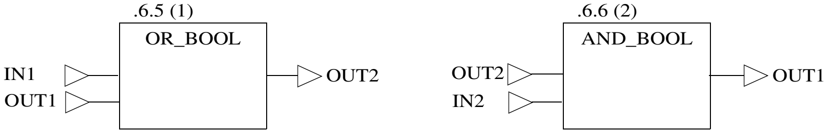
Resolved loop using an actual parameter: Variant 2



Resolution using Several Actual Parameters

Loops using several actual parameters are also allowed. With such loops, the execution order can later be influenced by executing – possibly several times – the menu command **Objects** → **Reverse FFB Execution Order** (see also *FFB Execution Order, p. 212*).

Loop using several actual parameters



7.4 Code generation with the FBD Function Block language

Code Generation Options

Introduction

Using the **Project** → **Code Generation Options** menu command, you can define options for code generation.

Include Diagnosis Information

If the **Include Diagnosis information** check box is checked, additional information for the process diagnosis (e.g. Transition Diagnosis (see *Transition diagnosis, p. 306*), diagnosis codes for diagnosis function blocks with extended diagnosis, such as e.g. XACT, XLOCK etc.) will be produced during code generation. This process diagnosis can be evaluated with MonitorPro or FactoryLink, for example.

Fastest Code (Restricted Checking)

If you check the **Fastest code (Restricted Checking)** check box, a runtime-optimized code is generated. This runtime optimization is achieved by realizing the integer arithmetic (e.g. "+" or "-") using simple CPU commands instead of EFB invocations.

CPU commands are much quicker than EFB invocations, but they do not generate any error messages, such as, for example, arithmetic or array overflow. This option should only be used when you have ensured that the program is free of arithmetic errors.

If **Fastest Code (Restricted Checking)** was selected, the addition IN1 + 1 is solved with the "add" CPU command. The code is now quicker than if the ADD_INT EFB were to be invoked. However, no runtime error is generated if "IN1" is 32767. In this case, "OUT1" would overrun from 32767 to -32768!

7.5 Online functions of the FBD Function Block language

Online Functions

Introduction

There are two animation modes available in the FBD editor:

- Animation of binary variables and links
- Animation of selected objects

These modes are also available on display of a DFB item (command button **Refine...** in the dialog box **Function block: xxx**).

Note: If the animated section is used as a transition section for SFC and the transition (and therefore also the transition section) is not processed, the status **DISABLED** appears in the animated transition section.

Animation of binary variables and links

The animation of binary variables and links is activated with the menu command **Online** → **Animate Booleans**.

In this mode, the current signal status of binary variables, direct addresses in the 0x and 1x range and binary links is displayed in the Editor window.

Animation of selected objects

The animation of the selected objects is activated with the menu command **Online** → **Animate selected**.

In this mode, the current signal status of the selected links, variables, multi-element variables and literals are displayed in the Editor window.

Note: If all variables/links of the section need to be animated, the whole section can be selected with **CTRL**+**A** and then **Online** → **Animate selected** (**CTRL**+**W**) all variables and links of the section will be animated.

If a numerical value is selected on an input/output, the name of the variable, its direct address and I/O assignment (if available) and its comment will be displayed in the status bar.

Note: The selected objects remain selected even after "Animate selected" has been selected again, in order to keep these for a further reading, and/or to be able to easily modify the list of objects.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in the Online help (Tip: Search the online help for the index reference "Colors").

7.6 Creating a program with the FBD Function Block language

Creating a Program in the FBD Function Block Language

Introduction

The following description contains an example for creating a program in the function block language (FBD). The creation of a program in the function block language is divided into 2 main steps:

Step	Action
1	Creating a Section (see *Creating a Section, p. 219*)
2	Creating the Logic (see *Creating the Logic, p. 220*)

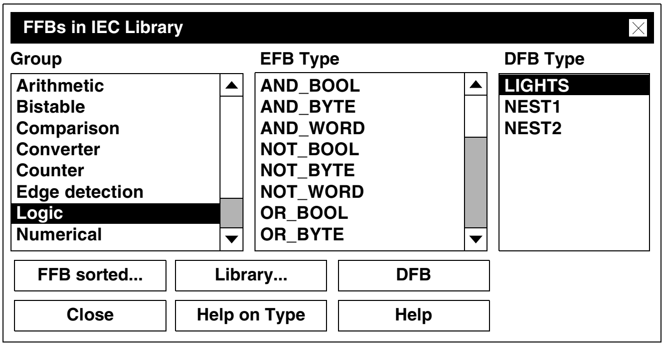
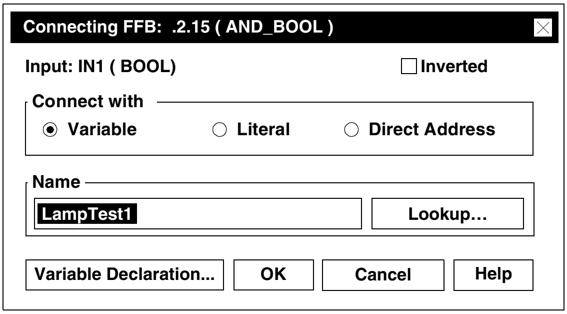
Creating a Section

The procedure for creating a section is as follows:

Step	Action
1	Using the **File** → **New Section...** menu command, create a new section and enter a section name. **Note:** The section name (max. 32 characters) is not case-sensitive and must be unique within the whole project. If the name entered already exists, you will be warned and you will have to choose a different name. The section name must comply with the IEC name conventions, otherwise an error message appears. **Note:**In compliance with IEC1131-3 only letters are permitted as the first character of names. However, if you wish to use numbers as the first character, you can enable this using the **Options** → **Preferences** → **IEC Extensions...** → **Allow Leading Digits in Identifiers** menu command.

Creating the Logic

The procedure for creating the logic is as follows:

Step	Action
1	To insert an FFB into the section, select the **Objects** → **Select FFB...** menu command. **Response:** The FFB dialog box from the library is opened.  FFBs in IEC Library ☒ Group: Arithmetic, Bistable, Comparison, Converter, Counter, Edge detection, Logic, Numerical EFB Type: AND_BOOL, AND_BYTE, AND_WORD, NOT_BOOL, NOT_BYTE, NOT_WORD, OR_BOOL, OR_BYTE DFB Type: LIGHTS, NEST1, NEST2 FFB sorted... | Library... | DFB Close | Help on Type | Help
2	In this dialog box you can select a library and an FFB from it by using the **Library...** command button. You can, however, also display the DFBs that you created and select one of them using the **DFB** command button.
3	Place the selected FFB in the section.
4	When all FFBs have been placed, close the dialog box with **Close**.
5	Activate the selection mode with **Objects** → **Select Mode**, click on the FFB and move the FFBs to the desired position.
6	Activate the link mode with **Objects** → **Link** and connect the FFBs.
7	Then re-activate select mode with **Objects** → **Select Mode** and double-click on one of the unconnected inputs/outputs. **Response:** The **Connect FFB** dialog box opens, where an actual parameter can be allocated to the input/output.  Connecting FFB: .2.15 (AND_BOOL) ☒ Input: IN1 (BOOL) □ Inverted Connect with: ● Variable ○ Literal ○ Direct Address Name: LampTest1 | Lookup... Variable Declaration... | OK | Cancel | Help

Step	Action
8	Depending on the program logic you can allocate the following to the input/output: ● **Variable** 　● Located variable 　You can allocate a hardware input/output signal to the input/output of the FFB using a located variable. 　The name of the variable is shown at the input/output in the editor window. 　● Unlocated variable 　You can use the unlocated variable allocated to the input/output of the FFB as a discrete, i.e. when resolving loops, or when transferring values between different sections. 　The name of the variable is shown at the input/output in the editor window. 　● Constant 　You can allocate a constant to the input of the FFB. The constant can be transferred to other sections. You determine the value of the constant in the variable editor. 　The name of the constant is shown at the input in the editor window. ● **Literal** You can allocate a literal to the input, i.e. directly allocate a value to the input/output. The value is shown at the input in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi element variables see *Calling Derived Data Types, p. 579*. **Note:** Unconnected FFB inputs are specified as "0" by default.
9	Save the FBD section with the menu command **File** → **Save Project** .

Ladder Diagram LD

8

At a Glance

Overview

This Chapter describes the Ladder Diagram LD which conforms to IEC 1131.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
8.1	General information about Ladder Diagram LD	225
8.2	Objects in Ladder Diagram LD	227
8.3	Working with the LD Ladder Diagram	243
8.4	Code generation with LD Ladder Diagram	249
8.5	Online functions with the LD Ladder Diagram	251
8.6	Creating a program withLD Ladder Diagram	253

8.1 General information about Ladder Diagram LD

General Information about the LD Ladder Diagram Language

Introduction

This section describes the Ladder Diagram (LD) according to IEC 1131-3.

The structure of a LD section corresponds to a rung for relay switching. The window in the LD editor is shaded with a logic grid, on the left side of which there is the so-called left power rail. This left power rail corresponds to the phase (L ladder) of a rung. With LD programming, in the same way as in a rung, only the LD objects (contacts, coils) which are linked to a power supply, that is to say connected with the left power rail, are "processed". The right power rail, which corresponds to the neutral ladder, is not shown optically. However, all coils and FFB outputs are linked with it internally and this creates a power flow.

Objects

The objects of the programming language LD (Ladder Diagram) help to divide a section into a number of:
- Contacts (see *Contacts, p. 228*),
- Coils (see *Coils, p. 230*) and
- FFBs (Functions and Function Blocks) (see *Functions and Function Blocks (FFBs), p. 233*).

These objects can be linked with each other through:
- Links (see *Link, p. 239*) or
- Actual Parameters (see *Actual Parameters, p. 240*).

Expansive logic can also be positioned in the LD section in the form of macros (related topics *Macros, p. 511*).

Theoretically, each section can contain as many FFBs and also as many inputs and outputs as required. It is therefore advisable to subdivide a whole program into logical units, that is to say into different sections.

Comments can be provided for the logic of the section with text objects (related topics *Text object, p. 242*).

Processing Sequence

Basically, LD sections are processed from top to bottom and from left to right.

Networks connected to the left power rail are processed from top to bottom.

The processing sequence of objects (contacts, coils, FFBs) is determined by the data flow within a network.

A detailed description can be found under *Execution sequence, p. 246*).

Editing with the Keyboard

Normally editing in Concept is performed with the mouse, however it is also possible with the keyboard (related topics *Shortcut keys in the LD-Editor, p. 847*).

In order to make editing with the keyboard easier, you can specify the number of columns per section in the CONCEPT.INI (see *INI Settings for the LD Section, p. 1125*) file, after which an automatic carriage return should appear when you are expanding a rung. This means that when you reach the last column, the next object is automatically placed in the second column of the next row. Objects on different rows are automatically linked, i.e. the objects are generated within a common rung.

IEC Conformity

For a description of the IEC conformity of the LD programming language see *IEC conformity, p. 857*.

8.2 Objects in Ladder Diagram LD

At a Glance

Overview

This section describes the objects in LD Ladder Diagram.

What's in this Section?

This section contains the following topics:

Topic	Page
Contacts	228
Coils	230
Functions and Function Blocks (FFBs)	233
Link	239
Actual Parameters	240
Text object	242

Contacts

At a Glance

A contact is an LD element that transfers a status on the horizontal link to its right side. This status comes from the boolean AND link of the status of the horizontal link on the left side, with the status of the relevant variable/direct address.

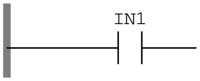
A contact does not change the value of the relevant variable/direct address.

The following contacts are available:
- Closer (see *Closer, p. 228*)
- Opener (see *Opener, p. 228*)
- Contact for detection of positive transitions (see *Contact for detection of positive transitions, p. 228*)
- Contact for detection of negative transitions (see *Contact for detection of negative transitions, p. 229*)

Closer

On closing, the status of the left link is copied onto the right link, if the status of the relevant boolean variable is ON. Otherwise, the status of the right link is OFF.

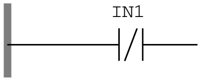
Closer



Opener

On opening, the status of the left link is copied onto the right link, if the status of the relevant boolean variable is OFF. Otherwise, the status of the right link is OFF.

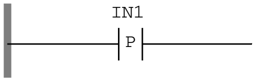
Opener



Contact for detection of positive transitions

With contacts for detection of positive transitions, the right link for a program cycle is ON if a transfer of the relevant boolean variable is made from OFF to ON and the status of the left link is ON at the same time. Otherwise, the status of the right link is OFF.

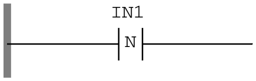
Contact for detection of positive transitions



Contact for detection of negative transitions

With contacts for detection of negative transitions, the right link for a program cycle is ON if a transfer of the relevant boolean variable is made from ON to OFF and the status of the left link is ON at the same time. Otherwise, the status of the right link is OFF.

Contact for detection of negative transitions



Coils

At a Glance

A coil is an LD element which transfers the status of the horizontal link on the left side, unchanged, to the horizontal link on the right side. The status is saved in the relevant variable/direct address.

Start behavior of coils

In the start behavior of PLCs there is a distinction between cold starts and warm starts:

- **Cold start**
  Following a cold start (load the program with **Load online** → **Load**) all variables (independent of type) are set to "0" or, if available, their initial value.
- **Warm start**
  In a warm start (stop and start the program or **Online** → **changes**) different start behaviors are valid for located variables/direct addresses and unlocated variables:
  - **Located variables/direct addresses**
    In a warm start all coils (0x registers) are set to "0" or, if available, their initial value.
  - **Unlocated variable**
    In a warm start all unlocated variables retain their current value (storing behavior).

This different behavior in a warm start leads to particular characteristics in the warm start behavior of LD objects "Coil – set" and "Coil – reset". Warm start behavior is dependent on the variable type used (storing behavior in use of unlocated variables; non storing behavior in use of located variables/direct addresses)

If a buffered coil is required with a located variable or with direct addresses, the RS or SR Function Block from the IEC block library should be used.

Available coils

The following coils are available:

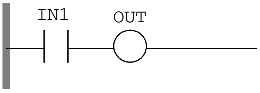
- Coil (see *Coil, p. 231*)
- Coil - negated (see *Coil - negated, p. 231*)
- Coil - set (see *Coil - set, p. 232*)
- Coil - reset (see *Coil - reset, p. 232*)
- Coil – positive edge (see *Coil – positive edge, p. 231*)
- Coil – negative edge (see *Coil – negative edge, p. 231*)

Coil

With coils, the status of the left link is copied onto the relevant Boolean variable and the right link.

Normally, coils follow contacts or EFBs, but they can also be followed by contacts.

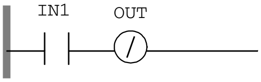
Coil



Coil - negated

With negated coils, the status of the left link is copied onto the right link. The inverted status of the left link is copied onto the relevant Boolean variable. If the left link is OFF, then the right link will also be OFF and the relevant variable will be ON.

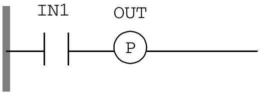
Coil - negated



Coil – positive edge

With coils for detection of positive transfers, the status of the left link is copied onto the right link. The relevant Boolean variable is ON for a program cycle, if a transfer of the left link from OFF to ON is made.

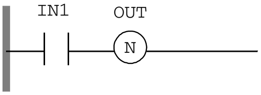
Coil – positive edge



Coil – negative edge

With coils for detection of negative transfers, the status of the left link is copied onto the right link. The relevant Boolean variable is ON for a program cycle, if a transfer of the left link from ON to OFF is made.

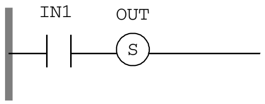
Coil – negative edge



Coil - set

With "set coils", the status of the left link is copied onto the right link. The relevant Boolean variable is set to ON status, if the left link is in ON status, otherwise it remains unchanged. The relevant Boolean variable can only be reset through the "reset coil".

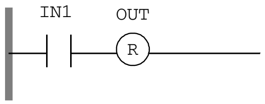
Coil - set



Coil - reset

With "reset coils", the status of the left link is copied onto the right link. The relevant Boolean variable is set to OFF status, if the left link is in ON status, otherwise it remains unchanged. The relevant Boolean variable can only be set through the "set coil".

Coil - reset



Functions and Function Blocks (FFBs)

Introduction

FFB is the generic term for:

- EFB (Elementary Function and Elementary Function Block) (see *EFB, p. 233*)
- DFB (Derived Function Block) (see *DFB, p. 236*)
- UDEFB (Derived Elementary Function and Derived Elementary Function Block) (see *UDEFB, p. 237*)

EFB

EFB is the generic term for:

- Elementary Function (see *Elementary Function, p. 234*)
- Elementary Function Block (see *Elementary Function Block, p. 235*)

EFBs are functions and function blocks that are available in Concept in the form of libraries. The logic of EFBs is built in C programming language and cannot be changed in the FBD editor.

Note: The EFBs AND_BOOL, NOT_BOOL, OR_BOOL, R_TRIG and F_TRIG are not available in LD. Their function is executed with contacts. The MOVE function cannot be used with the data type BOOL.

Elementary Function

Functions have no internal conditions. If the input values are the same, the value at the output is the same for all executions of the function. E.g. the addition of two values gives the same result at every execution.

An Elementary Function is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function, that is the function type, is displayed in the center of the frame. The function counter is displayed above the frame.

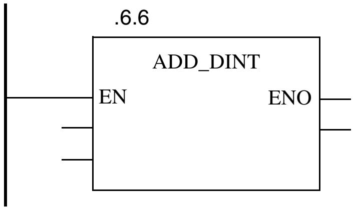
The function counter cannot be changed and always has an .n.m. structure.

.n = current section number

.m = current function number

Functions are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, p. 238*).

Elementary Function



Elementary Function Block

Function Blocks have internal conditions. If the inputs have the same values, the value at the output at every execution is another value. E.g. with a counter, the value on the output is incremented.

A function block is represented graphically as a frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function block, that is the function block type, is displayed in the center of the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

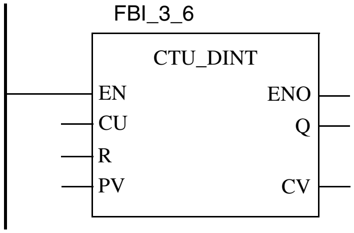
m = Number of the FFB object in the section (current number)

The instance name can be edited in the Properties dialog box of the function block. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears.

Note: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the **Options** → **Preferences** → **IEC Extensions...** → **Permit Leading Figures in Identifiers** menu command will enable this.

Function blocks are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, p. 238*).

Elementary Function Block



DFB

Derived Function Blocks are function blocks that have been defined in Concept DFB.

With DFBs, there is no distinction between functions and function blocks. They are always treated as function blocks regardless of their internal structure.

A DFB is represented graphically as a frame with double vertical lines and with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The DFB name is displayed centrally within the frame. The instance name is displayed above the frame. The instance name serves as a unique identification for the function block in a project.

The instance name is produced automatically with the following structure: FBI_n_m

FBI = Function Block Instance

n = Section number (current number)

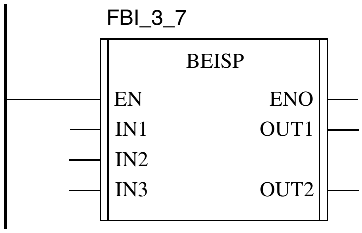
m = Number of the FFB object in the section (current number)

The instance name can be edited in the Properties dialog box of the DFB. The instance name must be unique throughout the whole project and is not case sensitive. If the name entered already exists, you will be warned and you will have to choose another name. The instance name must comply with the IEC name conventions otherwise an error message appears.

Note: In compliance with IEC1131-3 only letters are permitted as the first character of instance names. Should numbers be required as the first character however, the **Options** → **Preferences** → **IEC Extensions...** → **Permit Leading Figures in Identifiers** menu command will enable this.

Derived Function Blocks are only executed if the input EN=1 or if the input EN is grayed out (see also *EN and ENO, p. 238*).

Derived Function Block



UDEFB

UDEFB is the generic term for:
- User-defined Elementary Function
- User-defined Elementary Function Block

UDEFBs are functions and function blocks that have been programmed with Concept EFB in C++ programming language and are available in Concept in the form of libraries.

In Concept, there is no functional difference between UDEFBs and EFBs.

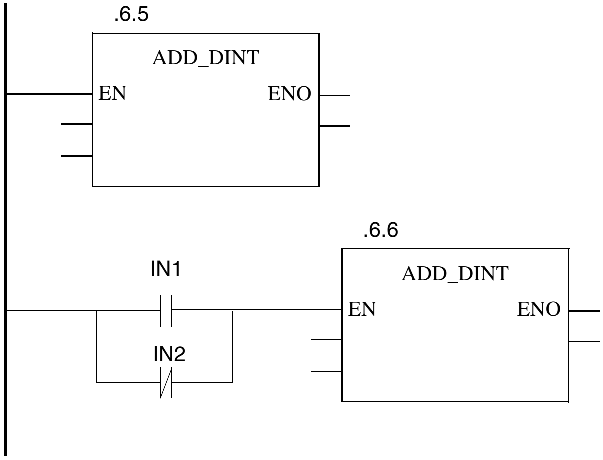
Editing FFBs

FFBs are only edited if at least one Boolean input is linked with the left power rail. If the FFB has no Boolean input, the EN input of the FFB must be used. If the FFB is to be conditionally executed, the Boolean input can be pre-linked through contacts or other FFBs.

Note: If the EN input is not linked with the left power rail, it must be deactivated in the Properties dialog box, otherwise the FFB will never be edited.

Note: Each FFB without Boolean link to the left power rail gives rise to an error message when downloading onto the PLC.

Connection to an FFB with the left power rail:



EN and ENO

With all FFBs, an EN input and an ENO output can be configured.

EN and ENO configuration is switched on or off in the FFB properties dialog box. The dialog box can be invoked with the **Objects** → **Properties...** menu command or by double-clicking on the FFB.

If the value of EN is equal to "0" when the FFB is invoked, the algorithms that are defined by the FFB will not be executed and all outputs keep their previous values. The value of ENO is automatically set to "0" in this case.

If the value of EN is equal to "1", when the FFB is invoked, the algorithms which are defined by the FFD will be executed. After successful execution of these algorithms, the value of ENO is automatically set to "1". If an error occurs during execution of these algorithms, ENO will be set to "0".

Note: If the EN input is not linked with the left power rail, it must be deactivated in the Properties dialog box, otherwise the FFB will never be edited.

The output behavior of the FFBs does not depend on whether the FFBs are invoked without EN/ENO or with EN=1.

Link

Description

Links are connections between contacts, coils and FFBs.

Several links can be connected with one contact, one coil or one FFB output. The link points are identified with a filled circle.

Note: Unconnected contacts, coils and FFB inputs are specified as "0" by default.

Data Types

The data types of the inputs/outputs to be linked must be the same.

Editing Links

Links can be edited in select mode. An overlap with other objects is permitted.

Configuring Loops

No loop can be configured with links because in this case, the execution order in the section cannot be determined uniquely. Loops must be resolved with actual parameters (related topics *Configuring Loops, p. 214*).

Horizontal Links

Contacts and coils are automatically connected during positioning with a neighboring, unconnected contact/coil that has the same vertical position. A connection to the power rail is only established if the contact is placed nearby (also see *Defining the Contact Connection, p. 1125* in the *Concept INI-File*chapter). If a coil or a contact is positioned on an existing horizontal link, the link is automatically separated and the contact/coil is inserted. When positioned, actual parameters may overlap another object, but they must not go outside the limits of the section frame. If a link to another object is established, this link is checked. If this link is not permitted, you will receive a message and the link will not be generated.

Once objects are positioned, horizontal links with directly adjacent objects are automatically created.

Vertical Links

An exceptional link is the "vertical link". The vertical link serves as a logical OR. With this form of the OR link, 32 inputs (contacts) and 64 outputs (coils, links) are possible.

Actual Parameters

Possible Actual Parameters

In the program runtime, the values from the process or from other actual parameters are transferred to the FFB via the actual parameters and then re-emitted after processing.

Table of possible actual parameters

Element	Actual Parameters
Contacts	● Direct addresses (see *Direct addresses, p. 46*) ● Located variables (see *Variables, p. 43*) ● Unlocated variable (see *Variables, p. 43*)
Coils	● Direct addresses (see *Direct addresses, p. 46*) ● Located variables (see *Variables, p. 43*) ● Unlocated variable (see *Variables, p. 43*)
FFB inputs	● Direct addresses (see *Direct addresses, p. 46*) ● Located variables (see *Variables, p. 43*) ● Unlocated variable (see *Variables, p. 43*) ● Constant (see *Constant variables, p. 44*) ● Literals (see *Literals (values), p. 45*)
FFB outputs	● Direct addresses (see *Direct addresses, p. 46*) ● Located variables (see *Variables, p. 43*) ● Unlocated variable (see *Variables, p. 43*)

Direct Addresses

The information on/display of direct addresses can be given in various formats. The display format is set in the **Options** → **Preferences** → **Common** dialog box. Setting the display format has no impact on the entry format, i.e. direct addresses can be entered in any format.

The following address formats are possible:
- **Standard Format (400001)**
  The five figure address comes directly after the first digit (the reference).
- **Separator Format (4:00001)**
  The first digit (the reference) is separated from the five figure address that follows by a colon (:).
- **Compact format (4:1)**
  The first digit (the Reference) is separated from the address that follows by a colon (:) where the leading zeros are not specified.
- **IEC Format (QW1)**
  There is an IEC type designation in initial position, followed by the five-character address.
  - %0x12345 = %Q12345
  - %1x12345 = %I12345
  - %3x12345 = %IW12345
  - %4x12345 = %QW12345

Data Types

The data type of the actual parameter must be of BOOL type with contacts and coils. With FFB inputs/outputs, the data type of the actual parameter must match the data type of the inputs/outputs. The only exceptions are generic FFB inputs/outputs, whose data type is determined by the formal parameter. If all actual parameters consist of literals, a suitable data type is selected for the function block.

Initial Values

FFBs, which use actual parameters on the inputs and coils that have not yet received a value assignment, work with the initial values of these actual parameters.

Unconnected Inputs

Note: Unconnected contacts, coils and FFB inputs/outputs are specified as "0" by default.

Text object

At a Glance

Text can be positioned in the form of text objects in the Ladder Diagram (LD). The size of these text objects depends on the length of the text. The size of the object, depending on the size of the text, can be extended vertically and horizontally to fill further grid units. Text objects may not overlap with other objects; however they can overlap with links.

Memory space

Text objects occupy no memory space on the PLC because the text is not downloaded onto the PLC.

8.3 Working with the LD Ladder Diagram

At a Glance

Overview

This section describes working with LD Ladder Diagram.

What's in this Section?

This section contains the following topics:

Topic	Page
Positioning Coils, Contacts, Functions and Function Blocks	244
Execution sequence	246
Configuring Loops	248

Positioning Coils, Contacts, Functions and Function Blocks

Positioning Objects

In the LD contact plan editor, the window has a logic grid in the background. The objects are aligned in the bars of this grid (52 x 230 fields) during positioning. With the exception of vertical shorts, FFBs and text fields, all elements require exactly one grid field. Objects can only be positioned within such a field. If an object is positioned between two fields, the object is automatically placed in the nearest field.

When objects are positioned outside the section frame with another object, an error message occurs and the object is not positioned.

When being positioned, contacts and coils are automatically linked with a directly adjacent, unconnected contact/coil, if the contact/ coil has the same vertical position. A link to the power rail is therefore created even if the contact is positioned 2 fields away. If contacts or coils are positioned on existing contacts or coils, the existing ones are replaced by the current ones (only applies to same types, i.e. when replacing coils with coils and contacts with contacts). If a coil or a contact is positioned on an existing horizontal short, the link is automatically separated and the contact/coil is inserted.

When positioned, actual parameters may overlap another object, but they must not go outside the limits of the section frame. If a link to another object is established, this link is checked. If this link is not permitted, you will receive a message and the link will not be generated. When producing links, overlaps and crossings with other links and objects are permitted.

If an FFB is selected, its comment is displayed in the first column of the status line. If an actual parameter is selected, its name and, if applicable, its direct address and its comment are displayed in the first column of the status line.

Automatic Carriage Return

As a keyboard user, you have the possibility of determining the number of columns/ fields in the CONCEPT.INI (see *Defining the Number of Columns/Fields, p. 1125*) file after which an automatic carriage return will appear during editing as soon as the last column/field is reached. The following object is then inserted into the second column/field and linked to the last object of the previous row. I.e. the objects are created inside the same rung.

Selecting FFBs

Using **Objects** → **Select FFB...** you can open a dialog for selecting FFBs. This dialog is modeless, which means it is not automatically closed once an FFB has been positioned, but remains open until you close it. If you have several LD sections open and you invoke the dialog, only one dialog box is opened and is available for all sections. The dialog box is not available for any other sections (not LD editor). If the LD sections are changed into symbols (Minimize window), the dialog box is closed. If one of the LD section symbols is invoked again, the dialog box is automatically re-opened.

The first time Concept is started, the FFB is displayed oriented to the library. This means that to select an FFB, the corresponding library must first be selected using the **Library** command button. Then you can select the corresponding group in the **Group** list box. Now, you can select the required FFB from the EFB type list box.

If you do not know which library/group the FFB required is located in, you can invoke an FFB-oriented dialog with the **FFB sorted** command button. This contains all FFBs in all libraries and groups in an alphabetical list.

After each subsequent project start, the view that you select will appear.

Once the FFB has been selected, its position in the section must be selected. The cursor becomes a small FFB and the cross shows the position (upper left corner of the FFB) in which the FFB is placed. The FFB is positioned by clicking on the left-hand mouse button.

Change FFB-Type

With the **Objects** → **Replace FFBs...** menu command, the FFBs already positioned in the section can be replaced with FFBs of another type (e.g. an AND with an OR). The variables given to the FFB remain if the data type and position of the inputs/outputs are the same in the "old" as the new FFB.

Note: FFBs with inputs/outputs of the ANY data type (generic FFBs) cannot be replaced.

Change contact/coil

Contacts and coils which are already positioned can simply be replaced. In order to do this, select the new element and click on the one to be replaced.

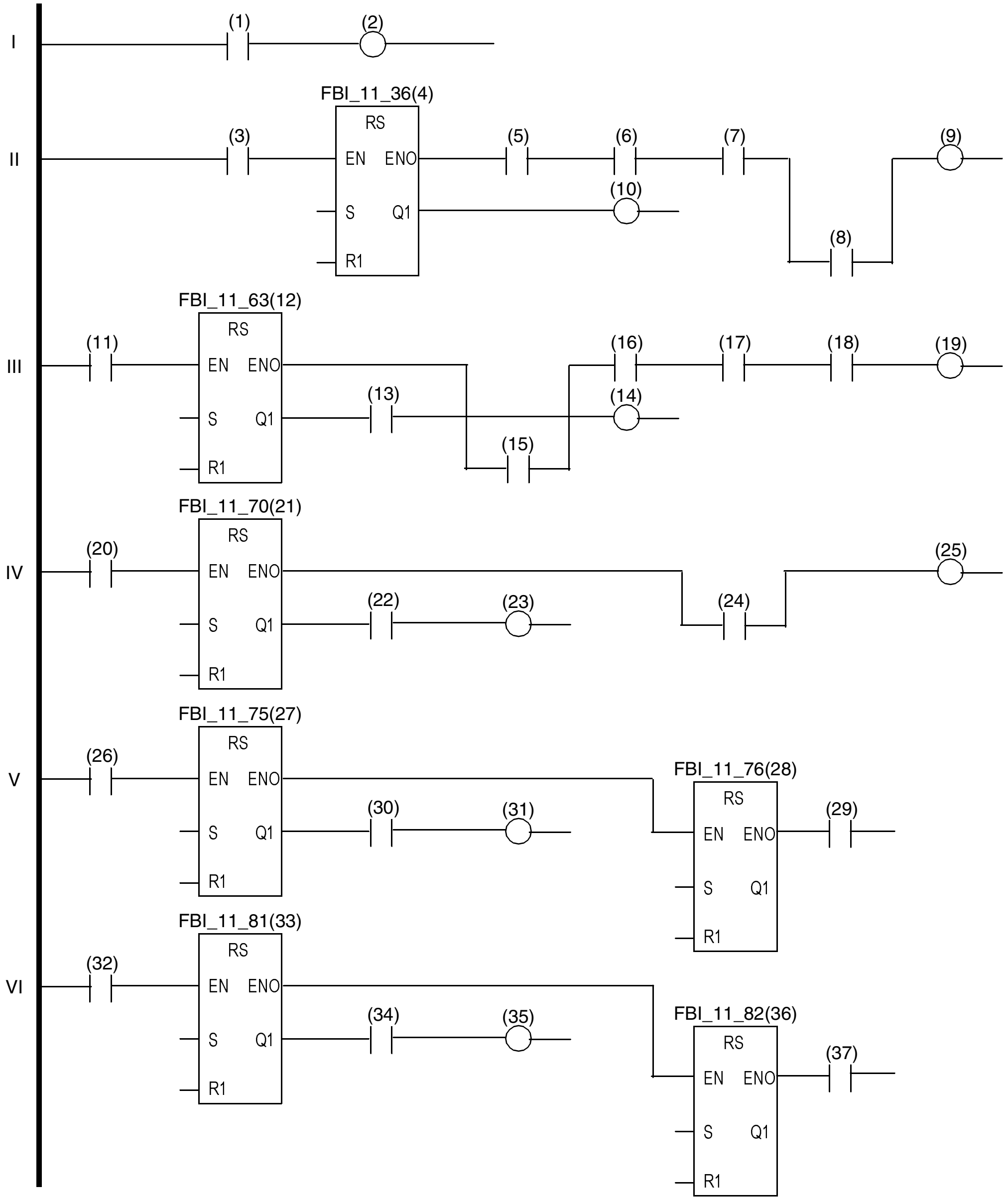
Execution sequence

Description

The following applies to the execution sequence in LD sections:

- The execution sequence of networks which are only linked by the left power rail, is determined by the graphic position in which the networks are connected to the left power rail.
  The networks are processed from top to bottom.
  See example below, Networks I-VI).
- The execution sequences of objects (contacts, coils FFBs) are determined by the data flow within a network. This means that the coils and FFBs whose inputs have already received value assignments will be processed first.
- Current paths that begin at outputs (Pins) from FFBs are processed according to the vertical, graphical position of its first object (from top to bottom).
  See example below, Network III):
  Processing after the FFB (FBI_11_63) begins with the current path whose first object is located at the uppermost vertical position (13) and thus follows current path (13)->(14).
  When current path (13)->(14) has been processed, processing of the next current path (15)->(19) begins.
- If the first objects of 2 current paths that begin at outputs (pins) of FFBs, at the same height, the first current path to be processed is that of the object that is farther left.
  See example below, Network IV): (22)->(23) then (24)->(25).
- The position of an FFB is determined by the upper left corner of the FFB.
  See example below.
  Network V: Upper left corner of FFB (FBI_11_76) above contact (30). Process: (28)->(29) then (30)->(31).
  Network VI: Upper left corner of FFB (FBI_11_82) same height as contact (34). Process: (34)->(35) then (36)->(37).

Example LD section

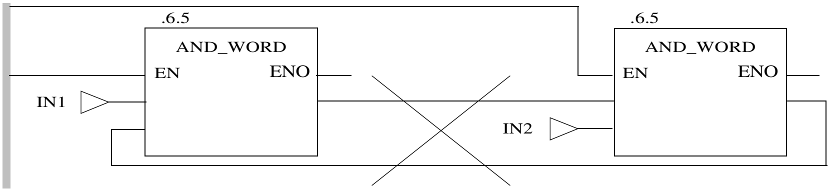


Configuring Loops

Non-permitted Loops

Configuring loops exclusively via links is not permitted, as it is not possible to make a unique specification of the data flow (the output of one FFB is the input of the next FFB, and the output of this one is the input of the first).

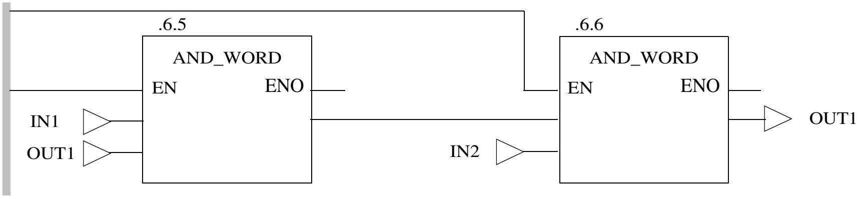
Non-permitted Loops via Links



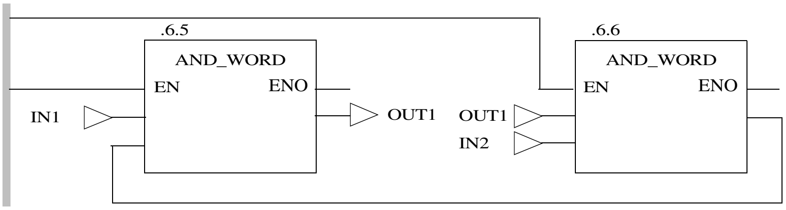
Resolution using an Actual Parameter

This type of logic must be resolved using actual parameters so that the data flow can be determined uniquely.

Resolved loop using an actual parameter: Variant 1



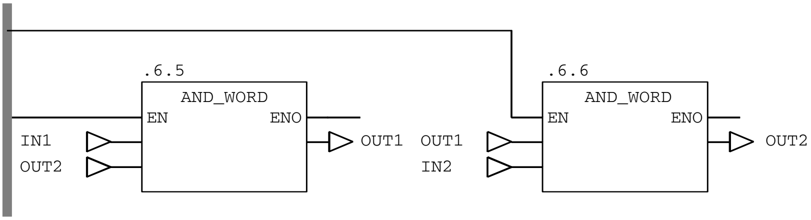
Resolved loop using an actual parameter: Variant 2



Resolution using Several Actual Parameters

Loops using several actual parameters are also allowed.

Loop using several actual parameters



8.4 Code generation with LD Ladder Diagram

Code Generation Options

Introduction

Using the **Project** → **Code Generation Options** menu command, you can define options for code generation.

Include Diagnosis Information

If you check the **Include Diagnosis Information** check box, additional information for the process diagnosis (e.g. transition diagnosis, diagnosis codes for diagnosis function blocks with extended diagnosis, such as XACT, XLOCK etc.) will be created during code generation. This process diagnosis can be evaluated with MonitorPro or FactoryLink, for example.

Fastest Code (Restricted Checking)

If you check the **Fastest code (Restricted Checking)** check box, a runtime-optimized code is generated. This runtime optimization is achieved by realizing the integer arithmetic (e.g. "+" or "-") using simple CPU commands instead of EFB invocations.

CPU commands are much quicker than EFB invocations, but they do not generate any error messages, such as, for example, arithmetic or array overflow. This option should only be used when you have ensured that the program is free of arithmetic errors.

If **Fastest Code (Restricted Checking)** was selected, the addition IN1 + 1 is solved with the "add" CPU command. The code is now quicker than if the ADD_INT EFB were to be invoked. However, no runtime error is generated if "IN1" is 32767. In this case, "OUT1" would overrun from 32767 to -32768!

8.5 Online functions with the LD Ladder Diagram

Online Functions

Introduction

There are two animation modes available in the LD editor:

- Animation of binary variables and links
- Animation of selected objects

These modes are also available when a DFB instance is displayed (command button **Refine...** in the **Function Block: xxx** dialog box).

Note: If the animated section is used as a transition section for SFC and the transition (and therefore also the transition section) is not processed, the status **DISABLED** appears in the animated transition section.

Animation of Binary Variables and Links

The animation of binary variables and links is activated using the **Online** → **Animate Booleans** menu command.

In this mode, the current signal status of binary variables, direct addresses in the 0x and 1x range and binary links is displayed in the editor window.

Meaning of Colors

Color	Meaning
Contact, coil, input/output, link red	Contact, coil, input/output, link transferring the value 0
Left power rail, contact, coil, input/output, link green	Left power rail, contact, coil, input/output, link transferring the value 1
Variable highlighted in beige	Variable forced
Variable highlighted in purple	Variable cyclically set
The name of the multi-element variable (e.g. motor) highlighted in color.	In the editor, a multi-element variable (e.g. motor) is displayed, in which one or more elements is forced or cyclically set.
The whole element name of the multi-element variable (e.g. right.motor.on) is highlighted in color.	In the editor, an element of a multi-element variable (e.g. right motor on) that is forced or cyclically set is displayed.
The name of the multi-element variable (e.g. right.motor.on) is highlighted in color, but the name of the element is not.	In the editor, an element of a multi-element variable (e.g. right motor on) that is not forced or cyclically set is displayed, but a different element of this multi-element variable is cyclically set or forced.

Animation of Selected Objects

The animation of the selected objects is activated with the **Online** → **Animate Selection** menu command.

In this mode, the current signal status of the selected links, variables, multi-element variables and literals is displayed in the editor window.

> **Note:** If you want to animate all variables/links in the section, you can select the whole section using **CTRL**+**A** and then animate all variables and links in the section using **Online** → **Animate Selection** (**CTRL**+**W**).

If a numerical value is selected on an input/output, the name of the variable, its direct address and I/O mapping (if existent) and its comment will be displayed in the status bar.

> **Note:** The selected objects remain selected even after "animate selection" has been selected again, to retain these objects for a further reading, and/or to be able to easily modify the list of objects.

Color key

There are 12 different color schemes available for animation. An overview of the color scheme and the meaning of each color can be found in the Online help Tip: Search the online hlep for the index reference "Colors").

8.6 Creating a program withLD Ladder Diagram

Creating a Program in LD

Introduction

The following description contains an example for creating a program in Ladder Diagram (LD). The creation of a program in LD Ladder Diagram is divided into 2 main steps:

Step	Action
1	Creating a Section (see *Creating a Section, p. 253*)
2	Creating the Logic (see *Creating the Logic, p. 254*)

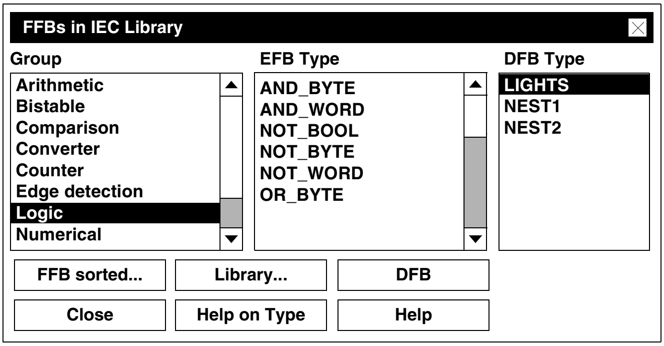
Creating a Section

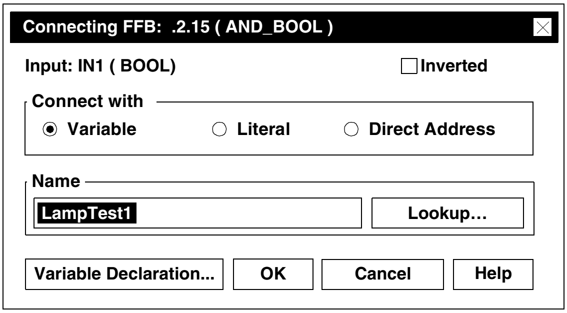
The procedure for creating a section is as follows:

Step	Action
1	Using the **File** → **New Section...** menu command, create a new section and enter a section name. **Note:** The section name (max. 32 characters) is not case-sensitive and must be unique within the whole project. If the name entered already exists, you will be warned and you will have to choose a different name. The section name must comply with the IEC name conventions, otherwise an error message appears. **Note:**In compliance with IEC1131-3 only letters are permitted as the first character of names. However, if you wish to use numbers as the first character, you can enable this using the **Options** → **Preferences** → **IEC Extensions...** → **Allow Leading Digits in Identifiers** menu command.

Creating the Logic

The procedure for creating the logic is as follows:

Step	Action
1	To insert a contact or coil in the section, open the **Objects** main menu and select the desired contact or coil. Contacts and coils can also be selected using the tool bar. Place the contact or coil in the section.
2	To insert an FFB into the section, select the **Objects** → **Select FFB...** menu command. **Response:** The **FFBs from Library** dialog box is opened.  FFBs in IEC Library Group Arithmetic Bistable Comparison Converter Counter Edge detection Logic Numerical EFB Type AND_BYTE AND_WORD NOT_BOOL NOT_BYTE NOT_WORD OR_BYTE DFB Type LIGHTS NEST1 NEST2 FFB sorted... Library... DFB Close Help on Type Help
3	In this dialog box you can select a library and an FFB from it by using the **Library...** command button. You can, however, also display the DFBs that you created and select one of them using the **DFB** command button.
4	Place the selected FFB in the section.
5	When all FFBs have been placed, close the dialog box with **Close**.
6	Activate select mode using **Objects** → **Select Mode**, and move the contacts, coils and FFBs to the required position.
7	Activate link mode with **Objects** → **Link**, and connect the contacts, coils and FFBs. Connect the contacts, FFBs and the left power rail.
8	Then re-activate select mode with **Objects** → **Select mode**, and double-click on a contact or coil. **Response:** The **Properties: LD objects** dialog box is opened, in which you can allocate an actual parameter to the contact/coil.

Step	Action
9	Depending on the program logic you can allocate the following to the contact/ coil: ● **Variable** ● **Located variable** You can allocate a hardware input/output signal to the input/output using a located variable. The name of the variable is shown at the input/output in the editor window ● **Unlocated variable** You can use the unlocated variable allocated to the input/output as a discrete, i.e. to resolve loops, or to transfer values between different sections. The name of the variable is shown at the input/output in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi-element variables see *Calling Derived Data Types, p. 579*. **Note:** Unconnected FFB inputs are specified as "0" by default.
10	To connect the FFB input/outputs to the actual parameters, double-click on one of the unconnected input/outputs. **Response:** The Connect FFB dialog box is opened, in which you can allocate an actual parameter to the input/output.  Connecting FFB: .2.15 (AND_BOOL) Input: IN1 (BOOL) □ Inverted Connect with ◉ Variable ○ Literal ○ Direct Address Name LampTest1 Lookup... Variable Declaration... OK Cancel Help

Step	Action
11	Depending on the program logic you can allocate the following to the input/ output: ● **Variable** ● Located variable You can allocate a hardware input/output signal to the input/output using a located variable. The name of the variable is shown at the input/output in the editor window ● Unlocated variable You can use the unlocated variable allocated to the input/output as a discrete, i.e. to resolve loops, or to transfer values between different sections. The name of the variable is shown at the input/output in the editor window. ● Constant You can allocate a constant to the input. The constant can be transferred to other sections. You determine the value of the constant in the variable editor. The name of the constant is shown at the input in the editor window. ● **Literal** You can allocate a literal to the input, i.e. directly allocate a value to the input/ output. The value is shown at the input in the editor window. ● **Direct address** You can allocate a hardware input/output signal to the input/output using an address. The address is shown at the input/output in the editor window. **Note:** For an example for invocation of multi-element variables see *Calling Derived Data Types, p. 579*. **Note:** Unconnected FFB inputs are specified as "0" by default.
12	Save the LD section using the **File** → **Save Project** menu command.

Index

Symbols

=> Assignment, 360, 429
•General information about online functions, 629
'SFCSTEP_STATE' variable, 264
'SFCSTEP_TIMES' variable, 263
'step'-variable, 264

A

Access Right, 774
Access Rights, 766, 775
Action, 265
Action variable, 265
Actions
  Process, 285
activate dialogs, 107
Actual parameters
  FBD, 205
actual parameters
  LD, 240
alias designations
  step, 293
  transition, 293
alternative branch, 273
Alternative connection, 275
Animation, 599, 753, 755
  FBD, 217
  General information, 679
  IEC section, 680
  IL, 373
Animation, 599, 753, 755
  IL/ST, 370
  LL984 section, 682
  Section, 679
animation
  LD, 251
  SFC, 298, 300
ANY Outputs, 426
Archiving
  DFB, 746
  EFB, 746
  Project, 746
ARRAY
  Range Monitoring, 583
ASCII message editor, 605, 607, 613
  Combination mode, 625
  Control code, 611
  Direct mode, 625
  Flush (buffer), 614
  Generals, 608
  How to continue after getting a warning, 623
  How to Use, 618
  Message Number, 619
  Message text, 620
  Offline mode, 625
  Repeat, 615
  Simulation text, 621
  Spaces, 612
  Text, 609
  User interface, 617, 618
  Variables, 610

ASCII messages, 63, 106
Assign instructions
    ST, 407
Assignment
    =>, 429
Atrium
    Memory optimization, 189
Atrium example
    INTERBUS controller, 963
Atrium first startup
    DOS Loader, 1100
    EXECLoader, 1082
    Modbus Plus, 1082, 1100
Atrium INTERBUS controller, 964
Auto-Log-Out, 133
Automatic Connection, 1160, 1163
available functions in OFFLINE and ONLINE modes, 85

B

Backplane Expander
    Edit I/O Map, 119
    Error handling, 120
    Generals, 118
Backplane Expander Config
    Configure, 117
Block call up
    IL, 355
    ST, 425

C

Call
    FFB, 362
    Project, 817
call
    DFB, 352
    FFB, 352
Chain jump, 272
Chain loop, 272
change
    coil, LD, 245
    contact, LD, 245
    FFB, FBD, 211
    FFB, LD, 245
Changing signal states of a Located variable
    Reference data editor, 592
Close Column
    LL984, 450
Closer
    LD, 228
Code generation
    IL, 368
    ST, 433
code generation
    FBD, 215
    LD, 249
coil
    change, LD, 245
    replace, LD, 245
Coil - negated
    LD, 231
Coil – negative edge
    LD, 231
Coil – positive edge
    LD, 231
Coil - reset
    LD, 232
Coil - set
    LD, 232
Coils
    LD, 230
Cold restart, 44
Comments
    Data type editor, 576
    Derived data type, 576
Communication, 18
Compact
    Memory optimization, 173
Compact configuration
    RTU extension, 127
Compact example, 957
Compact first startup
    DOS Loader, 1063, 1097
    EXECLoader, 1048, 1078
    Modbus, 1048, 1063
    Modbus Plus, 1078, 1097
Concept DFB, 469, 511
Concept M
    Hardware Package Contents, 7

Concept ModConnect, 1005
- Integrating new Modules, 1009
- Removing modules, 1010
- Use of Third Party Modules in Concept, 1011

Concept PLCSIM32, 755
Concept S
- Hardware Package Contents, 7

Concept Security, 763, 764, 766, 774, 775, 776
Concept SIM, 753
Concept XL
- Hardware Package Contents, 7

CONCEPT.INI, 1115, 1117
- Exclusion of the global/local DFBs from Online-Backup, 1121
- general, 1118
- LD section settings, 1125
- Path for global DFBs, 1121
- Path for Help Files, 1121
- print settings, 1119
- Project Name Definition, 1120
- Reading global DFBs, 1121
- Register Address Format Settings, 1120
- representation of internal data, 1124
- Saving the global DFBs during upload, 1121
- Security Settings, 1128
- Setting for Online Processing, 1126
- Setting for the Address Format, 1127
- Settings for the RDE, 1129
- Settings for the Tools menu, 1130
- Settings for Warning Messages, 1127
- Variable Storage Settings, 1120

Configuration, 77
- General information, 79
- Optional, 105
- Unconditional, 87

Configuration example
- Atrium-INTERBUS controller, 963
- Compact controller, 957
- Momentum-Ethernet bus system, 985
- Momentum-Remote I/O bus, 973
- Quantum-INTERBUS control, 919

Configuration example
- Quantum-Peer Cop, 949
- Quantum-Profibus DP controller, 933
- Quantum-Remote control with DIO, 909
- Quantum-Remote control with RIO, 889
- Quantum-Remote control with RIO (series 800), 897
- Quantum-SY/MAX controller, 925

configuration extensions, 107
Configuration in OFFLINE and ONLINE mode, 83
- General information, 84

Configuration of various network systems, 121
Configurator
- Ethernet I/O Scanner, 128

Configure
- Backplane Expander Config, 117
- INTERBUS, 122
- RTU extension, 127

configure, 59
- Ethernet, 125
- Profibus DP, 123

configure Ethernet, 125
Configure INTERBUS system, 122
Configure network systems, 121
configure network systems, 107
configure Profibus DP system, 123
Connect
- PLC, 631

Connect to PLC, 631
connecting IEC Simulator (32-bit), 645
connecting to IEC Simulator (32-bit), 645
Connecting to the PLC
- automatically with command line parameters, 1160
- Automatically with the CCLaunch Tool, 1163

Constant Scan, 649
Constants, 42
contact
- change, LD, 245
- replace, LD, 245

Contacts
- LD, 228, 229

Context help, 819
Convert
  DFBs, 1001
  Macros, 1001
  Projects, 1001
converting RDE templates, 590
Convertion
  Modsoft programs, 1013
CPU selection for the PLC type, 90
Create
  DFB, 491
  Macro, 523
  Program, 55
  Project, 55
  Project Symbol, 817
create
  FFB, FBD, 210
  FFB, LD, 244
Creating a program
  IL, 375
Cyclical Setting of Variables
  Reference Data Editor, 594

D

data exchange between nodes on the Modbus Plus network, 108
data flow, 248
  FBD, 214
Data Protection, 62
data protection in the state RAM, 111
Data Type Definition
  Extended (larger than 64 Kbytes), 565
Data Type Editor
  Short Cut Keys, 841
Data type editor, 557, 559
  Comments, 576
  Elements, 568
  Syntax, 567
data type editor, 560
  key words, 569
  separators, 574
  use of memory, 577
Datatype editor
  Names, 573
DDT, 565
Declaration of variables, 537
Declare
  Actions, 285
  Step properties, 283
declare
  transition, 291
Declare Variables, 537
Defining Colors
  INI File, 1126
defining the LD contact connection
  settings in the INI file, 1125
defining the number of LD columns/fields
  settings in the INI file, 1125
Delete
  DFB, 749
  Macro, 749
  Project, 749
Deleting memory zones from the PLC, 652
Deleting PLC contents, 652
Derived Data Type, 559
  Elements, 568
  Export, 697
  Syntax, 567
Derived data type, 557
  Comments, 576
  Names, 573
derived data type, 560
  global, 563
  key words, 569
  local, 563
  separators, 574
  use of memory, 577
Derived Data Types
  Use, 579
Derived Function Block, 472
derived function block
  LD, 236
Derived Function Blocks
  FBD, 202
DFB, 469, 472
  Archiving, 746
  call, 352
  context sensitive help, 488
  Create, 491
  Creating Global Variables, 484

DFB, 469, 472
  Delete, 749
  Documentation, 735
  FBD, 202
  global, 474
  invocation, 356, 426
  LD, 236
  local, 474
  Protect, 776
DFBs
  Convert, 1001
Diagnosis
  Transition diagnosis, 306
diagnosis viewer, 683
diagnostics viewer, 683
Dialog boxes, 815
Dialog interaction
  LL984, 445
Direct Addresses, 42
Disable
  Interrupt Sections, 49
  Section, 49
document section options, 739
Documentation
  Contents, 736
  DFB, 735
  Keywords, 743
  Macro, 735
  Project, 735
documentation
  layout, 737
DOS Loader
  Atrium first startup, 1100
  Compact first startup, 1063, 1097
  Momentum first startup, 1066, 1069, 1103, 1106
  Quantum first startup, 1060, 1094
  Startup when using Modbus, 1059
  Startup when using Modbus Plus, 1093
Download Changes, 671
Downloading Changes, 671
Driver for 16 bit application capability with Windows 98/2000/NT
  Virtual MBX Driver, 1031
Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/2000/NT
  MBX-Treiber, 1032
Driver for Modbus Plus Function via TCP/IP
  Ethernet MBX Driver, 1034
Driver for Remote Operation
  Remote MBX Driver, 1033
DTY, 557, 559, 560
DX Zoom
  LL984, 452

E

Edit
  Actions, 285
  LL984, 444, 449
  SFC, 280
  Step properties, 283
edit
  SFC, 279
  transition, 291
Edit I/O Map
  Backplane Expander, 119
Editing local Drop, 890
Editing Networks
  LL984, 450
Editors, 13
EFB
  Archiving, 746
  FBD, 200
  LD, 233
EFBs for Interrupt Sections, 1157
Elementary Function
  FBD, 200
elementary function
  LD, 234
Elementary Function Block
  FBD, 201
elementary function block
  LD, 235
Elements
  Data type editor, 568
  Derived Data Type, 568

EN
FBD, 203
LD, 238
ENC File, 19, 685
Encoded Log, 19
Encrypt Logfile, 765
Encrypted Logging
ENC File, 685
ENO
FBD, 203
LD, 238
EQUAL, 632
Equation network
LL984, 457, 458
Equation network, Syntax and Semantics
LL984, 462
Error handling
Backplane Expander, 120
Establishing the hardware connection
Modbus Plus presettings, 1036
Modbus presettings, 1041
Ethernet, 644
Ethernet / I/O Scanner
How to use the Ethernet / I/O Scanner, 131
Ethernet Bus System
Create online connection, 999
Ethernet Bus System (Momentum), 986
Ethernet I/O Scanner
Configurator, 128
Ethernet MBX Driver
Driver for Modbus Plus Function via TCP/IP, 1034
ethernet with Momentum, 126
Ethernet with Quantum, 125
Event Viewer
INI Settings, 1133
Example of hardware configuration
Atrium-INTERBUS controller, 963
Compact controller, 957
Momentum-Ethernet bus system, 985
Momentum-Remote I/O bus, 973
Quantum-INTERBUS control, 919
Quantum-Peer Cop, 949
Example of hardware configuration
Quantum-Profibus DP controller, 933
Quantum-Remote control with DIO, 909
Quantum-Remote control with RIO, 889
Quantum-Remote control with RIO (Series 800), 897
Quantum-SY/MAX controller, 925
Exchange Marking
Macro, 518
Exclusion of the global/local DFBs from Online-Backup
Settings in the INI-File, 1121
EXEC file
CPU 424 02, 150
CPU X13 0X, 150
Momentum, 186
EXEC files, 1109
EXECLoader
Atrium first startup, 1082
Compact first startup, 1048, 1078
Momentum first startup, 1051, 1055, 1085, 1089
Quantum first startup, 1044, 1074
Startup when using Modbus, 1043
Startup when using Modbus Plus, 1073
Execution Order
Timer Event Sections, 1145
Execution order
Section, 48
execution order
FBD, 212
Execution sequence
LD, 246
Export, 689
Derived Data Type, 697
General Information, 691
PLC Configuration, 731
Section, 693
Variable, 697
export
PLC configuration, 730
Exporting located variables, 547
Expressions
ST, 381
extended memory, 154

F

Factory Link, 726
FBD, 195
  Actual parameters, 205
  Animation, 217
  Calling a macro, 532
  code generation, 215
  data flow, 212, 214
  Derived Function Blocks, 202
  DFB, 202
  EFB, 200
  Elementary Function, 200
  Elementary Function Block, 201
  EN, 203
  ENO, 203
  execution order, 212
  FFB, 200
  Function, 200
  Function Block, 201
  Icon bar, 829
  link, 204
  loop, 214
  Online Functions, 217
  program creation, 219
  Short Cut Keys, 843
  Text Object, 207
  UDEFB, 203
  User-defined Elementary Function, 203
  User-defined Elementary Function Block, 203
FFB
  Call, 362
  call, 352
  change, FBD, 211
  change, LD, 245
  create, FBD, 210
  create, LD, 244
  FBD, 200
  insert, FBD, 210
  insert, LD, 244
  invocation, 356, 426, 430
  LD, 233
  position, 210, 244
  replace, FBD, 211
  replace, LD, 245
forcing
  I/O, 588
Function
  FBD, 200
function
  LD, 234
Function Block
  FBD, 201
function block
  LD, 235
Function Block language, 195
Function Blocks for Interrupt Sections, 1157

G

General, 1
  PLC Connection, 632
General information
  Loading a project, 668
  Online functions, 629
  Select process information, 662
General information about configuration in OFFLINE and ONLINE mode, 84
General information about hardware configuration, 79
general information about the online control panel, 648
General information about the PLC configuration, 80
General Information about the Reference Data Editor, 588
General to the variables editor, 536
Generals
  Backplane Expander, 118
Generate
  Project symbol, 817
Global data transfer
  Peer Cop, 953
global derived data type, 563
global DFB, 474
Global DFBs
  Defining the Path, 1121
  INI File, 1121
  Reading, 1123
  Storing, 1122

global macro, 516
Global Variables in DFBs, 484

H

hardware
- performance, 781

Head setup, 61
Help, 819
Help Files
- Defining the Path, 1121

How to use the Ethernet / I/O Scanner
- Ethernet / I/O Scanner, 131

I

I/O
- forcing, 588

I/O Event Section
- Handling, 1135

I/O Event Sections, 1154
- Priority, 1155
- Runtime Error, 1156

I/O map, 60, 101
Icon bar, 827, 828, 829, 830, 832
Icons, 825, 827, 828, 829, 830, 832, 833, 834, 837
icons, 835, 836
Icons_Project Browser, 837
Identifier, 288
IEC
- Momentum first startup, 1051, 1085, 1103

IEC conformity, 857
IEC Hot Standby data, 95
IEC section
- Animation, 680

IL, 307
- Animation, 370, 373
- Block call up, 355
- Code generation, 368
- Creating a program, 375
- Instruction, 311, 312
- List of Symbols, 833
- Modifier, 316

IL, 307
- Online functionen, 373
- Online functions, 369, 370
- Operands, 314
- Operators, 318, 327
- Short Cut Keys, 841
- syntax check, 366
- Tag, 321

IL Command
- Comments, 325
- Compare, 345, 346, 348
- Declaration, 323

IL command
- call function block, 356
- Compare, 343, 344, 347
- DFB invocation, 356
- invert, 338
- Reset, 331
- Set, 330
- VAR...END_VAR, 323

IL operation
- addition, 339
- Boolean AND, 332
- Boolean exclusive OR, 336
- Boolean OR, 334
- call DFB, 352
- call function block, 352
- jump to label, 349
- Load, 328
- multiplication, 341
- Store, 329
- subtraction, 340

IL-Befehl
- Function call, 362

IL-operation
- division, 342

Import, 689
- General Information, 691
- Multiple Address Assignment, 727
- PLC Configuration, 731
- Section, 699, 705, 714, 715, 716
- Structured variables, 723
- Variables, 719, 723, 726

import
PLC configuration, 730
section, 700, 711
variables, 720
import INTERBUS configuration, 970
importing Profibus DP configuration, 941
INC
Include File, 565
Include File
Extended Data Type Definition, 565
INI File
Event Viewer Settings, 1133
General Information, 1132
Project Name Definition, 1120
Project Specific, 1115
Register Address Format Settings, 1120
Security Settings, 1128
Setting for Online Processing, 1126
Settings for the Address Format, 1127
Settings for Warning Messages, 1127
Variable Storage Settings, 1120
INI file
general, 1118
LD section settings, 1125
print settings, 1119
representation of internal data, 1124
Settings for the Tools menu, 1130
INI Files, 1115
CONCEPT.INI, 1117
Projectname.INI, 1131
INI-File
Exclusion of the global/local DFBs from Online-Backup, 1121
Reading global DFBs, 1121
Settings for the Online-Backup, 1134
Settings for the RDE, 1129
INI-file
Path for global DFBs, 1121
Path for Help files, 1121
Saving the global DFBs during upload, 1121
initial step, 262
insert
FFB, FBD, 210
FFB, LD, 244
Install loadables, 60
installing the EXEC file, 1109
installing the Modbus Plus driver
Windows 98/2000/NT, 1030
Installing the SA85/PC185
Modbus Plus Preferences, 1028
Windows NT, 1028
Installing the SA85/PCI85
Modbus Plus Preferences, 1024
Windows 98/2000/XP, 1024
Instruction
IL, 311, 312
Instruction list, 307
Instructions
ST, 408
INTERBUS controller, 920
INTERBUS export settings in CMD, 965
Interface Settings in Windows 98/2000/XP
Modbus Preferences, 1038
Interface settings in Windows NT
Modbus Presettings, 1040
Interrupt Processing, 1135
General, 1137
Interrupt Sections
Disable, 49
EFBs, 1157
Examples for Setting Parameters, 1149
Execution Order, 1145
I/O Event Sections, 1154
Operating System, 1146
Priority, 1155
Runtime Error, 1156
Scan Rate for Timer Event Sections, 1141
Timer Event Sections, 1140, 1142
invocation
DFB, 356, 426
FFB, 356, 426, 430
Invoke
Project, 817

J

Jump
SFC, 272

K

Key combinations, 825, 839, 840, 841, 843, 847, 854
key words
- data type editor, 569
- derived data type, 569

Keys, 825, 839, 840, 841, 843, 847, 854

L

Ladder Diagram, 223
Ladder Logic 984, 439
LD, 223
- actual parameters, 240
- animation, 251
- Calling a macro, 532
- Closer, 228
- code generation, 249
- Coil - negated, 231
- Coil – negative edge, 231
- Coil – positive edge, 231
- Coil - reset, 232
- Coil - set, 232
- Coils, 230
- Contacts, 228, 229
- Data flow, 246
- derived function block, 236
- EFB, 233
- elementary function, 234
- elementary function block, 235
- EN, 238
- ENO, 238
- Execution sequence, 246
- FFB, 233
- function, 234
- function block, 235
- Icon bar, 832
- link, 239
- loops, 248
- online functions, 251
- Opener, 228
- program creation, 253
- Shortcut keys, 847
- Text object, 242

LD, 223
- UDEFB, 237
- user-defined elementary function, 237
- user-defined elementary function block, 237

Learn monitoring times
- SFC, 303

Libraries, 11
Limitations
- LL984, 442

link
- FBD, 204
- LD, 239

List of Symbols, 833, 834
List of symbols, 825
List of Tools, 833, 834
List of tools, 825
Literals, 42
LL984, 439
- Close Column, 450
- Combination mode, 467
- Dialog interaction, 445
- Direct programming, 467
- DX Zoom, 452
- Edit, 444, 449
- Editing Networks, 450
- Equation network, 457, 458
- Equation network, Syntax and Semantics, 462
- List of Symbols, 834
- Momentum first startup, 1055, 1069, 1089, 1106
- Navigation, 444
- Online Restriction, 445
- Online Search, 453
- Open Column, 450
- Open Row, 450
- Programming modes, 467
- Reference Offset, 448
- Reference Zoom, 451
- References, 446
- Replace References, 453
- Requirements, 444
- Section, 441
- Segement, 441

LL984, 439
  Select, 449
  Short Cut Keys, 854
  Subroutines, 455
  Trace, 453
  Undo, 449
  Variables, 446
LL984 Processing
  speed optimized, 653
LL984 section
  Animation, 682
Load reference data, 603
Loadables
  CPU 424 02, 156
  CPU X13 0X, 156
loadables, 96
  Atrium, 191
  compact, 176
  CPU 434 12, 164
  CPU 534 14, 164
Loading, 669
Loading a project, 667
  General information, 668
loading firmware, 1109
local derived data type, 563
local DFB, 474
local macro, 516
Located variables
  Changing signal states in RDE, 592
Log Encoding, 19
LOG File, 685
Logging
  LOG File, 685
Logging Write Access to the PLC, 685
loop
  FBD, 214
loops
  LD, 248

M

Macro, 511, 514
  Calling up from SFC, 529
  Calls from FBD, 532
  Calls from LD, 532
  Create, 523
Macro, 511, 514
  Delete, 749
  Documentation, 735
  Exchange marking, 518
macro
  context sensitive help, 521
  global, 516
  local, 516
Macros
  Convert, 1001
maximum supervision time, 262
MBX Driver
  Driver for connection between ModConnect Host interface adapters and 32 bit applications with Windows 98/ 2000/NT, 1032
Memory, 137
  Optimize, 142
  PLC-Independent Memory Optimization, 143
  Structure, 139
Memory and optimization
  Atrium, 189
  Compact, 173
  Momentum, 183
  Quantum, 147, 161
Memory partitions, 59
Memory statistics, 664
Menu commands, 813
minimum configuration, 59
minimum supervision time, 263
MMS-Ethernet
  specify coupling modules, 107
Modbus
  Compact first startup, 1048, 1063
  Momentum first startup, 1051, 1055, 1066, 1069
  Quantum first startup, 1044, 1060
  Startup with DOS Loader, 1059
  Startup with the EXECLoader, 1043
Modbus communication, 61
Modbus network link, 636
Modbus Plus
  Atrium first startup, 1082, 1100
  Compact first startup, 1078, 1097

Modbus Plus
- Momentum first startup, 1085, 1089, 1103, 1106
- Quantum first startup, 1074, 1094
- Remote MBX Driver, 1033
- Startup with DOS Loader, 1093
- Startup with the EXECLoader, 1073
- Virtual MBX Driver, 1031
- Write Restriction, 133

Modbus Plus Bridge, 642

Modbus Plus Network Connection, 637

Modbus Plus network node, 108

Modbus Plus Preferences
- Installing the SA85/PC185, 1028
- Installing the SA85/PCI85, 1024

Modbus Plus preferences
- installing the Modbus Plus driver in Windows 98/2000/NT, 1030

Modbus Plus presettings
- Establishing the hardware connection, 1036
- Startup, 1023

Modbus Plus Routing Path
- Automatic Connection, 1160, 1163

Modbus Preferences
- Interface Settings in Windows 98/2000/XP, 1038

Modbus Presettings
- Interface Settings in Windows NT, 1040
- Transfer problems, 1042

Modbus presettings
- Establishing the hardware connection, 1041
- Startup, 1037

ModConnect, 1005

MODIFIED, 632

Modifier
- IL, 316

Modsoft
- Convertion, 1013
- Function compatibility, 1022
- References, 1019

Momentum
- Memory optimization, 183

Momentum example
- Ethernet bus system, 985
- Remote I/O bus, 973

Momentum first startup
- DOS Loader, 1066, 1069, 1103, 1106
- EXECLoader, 1051, 1055, 1085, 1089
- Modbus, 1051, 1055, 1066, 1069
- Modbus Plus, 1085, 1089, 1103, 1106

MSTR-Read-Operation, 134

N

Names
- Datatype editor, 573
- Derived datatype, 573

Navigation
- LL984, 444

Network Configuration
- TCP/IP, 987

Network Connection
- Modbus Plus, 637

network link
- Modbus, 636
- TCP/IP, 644

NOM/NOE
- Disable Write Access, 133

NOT EQUAL, 632

O

Objects
- SFC, 261

objects
- insert, LD, 244

offline functions in the configurator, 85

Online, 753, 755
- INI File, 1126
- SFC, 297

Online Control Panel, 654, 658

online control panel
- general information, 648

online diagnosis, 683

online diagnostics, 683

Online functionen
- IL, 373

Online Functions
- FBD, 217

Online functions, 17
- General information, 629
- IL, 369
- IL/ST, 370
- ST, 435

online functions
- LD, 251
- SFC, 298, 300

online functions in the configurator, 85

Online help, 819

ONLINE Operation
- Presettings, 635

Online Restriction
- LL984, 445

Online Search
- LL984, 453

Online-Backup
- INI-Settings, 1134

Online-Funktionen, 627

Open
- Project, 817

Open Column
- LL984, 450

Open Row
- LL984, 450

Opener
- LD, 228

Operands
- IL, 314
- ST, 382

Operating System
- Timer Event Sections, 1146

Operators
- IL, 318, 327
- ST, 387

operators
- ST, 384

Optimize
- PLC Memory, 142
- PLC-Independent Memory Optimization, 143

Optional Configuration, 105

P

page breaks for sections, 739

Parallel branch, 276

Parallel connection, 277

Parameterize ASCII interface, 112

Parameterize interfaces
- ASCII interface, 112
- Modbus interface, 112

Parameterize Modbus interface, 112

Parameters for Automatic Connection, 817

Password Protection, 764, 766, 774, 775

Password protection, 763

Path for global DFBs
- Settings in the INI-File, 1121

Path for Help files
- Settings in the INI-File, 1121

Peer Cop, 108, 950

Peer Cop communication, 62

performance
- hardware, 781
- PLC family, 781

Phase
- Timer Event Sections, 1142

PLC
- Simulating, 751
- Status, 810

PLC Configuration
- Export, 731
- Import, 731

PLC configuration, 58, 59, 77
- export, 730
- General information, 80
- icons, 835
- import, 730

PLC Connection
- General, 632

PLC family
- performance, 781

PLC Hardware Package Contents in Concept S, M and XL, 7

PLC Memory, 137
- Optimize, 142
- PLC-Independent Memory Optimization, 143
- Structure, 139

PLC Memory and optimization
- Atrium, 189
- Compact, 173
- Momentum, 183
- Quantum, 147

PLC memory and optimization
- Quantum, 161

PLC memory mapping, 95
PLC selection, 89
PLC State, 646
PLC state, 663
PLC Status, 632
position
- FFB, FBD, 210
- FFB, LD, 244

precondition for unconditional configuration, 88
Presettings for Modbus
- Startup, 1037

Presettings for Modbus Plus
- Startup, 1023

Presettings for ONLINE operation, 635
print
- settings in the INI file, 1119

printing sections, 739
Priority
- I/O Event Sections, 1155

proceed in the following way with the configuration, 81
Process
- Actions, 285
- Step properties, 283

process
- transition, 291

Processing
- Program, 36
- Project, 36

PROFIBUS
- specify coupling modules, 107

Profibus DP controller, 934
Profibus DP export settings in SyCon, 935
Program
- Create, 55
- Processing, 36
- Status, 810
- Structure, 35, 36

Program creating
- ST, 437

program creation
- FBD, 219
- LD, 253

Programming, 9
Programming languages, 13
Programming modes
- LL984, 467

Programs, 42
Project
- Archiving, 746
- Call, 817
- Create, 55
- Delete, 749
- Documentation, 735
- Invoke, 817
- Open, 817
- Processing, 36
- Protect, 776
- Structure, 35, 36

Project Browser, 549
- Keyboard operation, 554
- Mouse operation, 554
- Toolbar, 837

Project Name Definition
- INI File Settings, 1120

Project Symbol
- Create, 817

Project symbol
- Generate, 817

Projectname.INI, 1115, 1131
- Event Viewer Settings, 1133
- General Information, 1132

Projects
- Convert, 1001

Projektname.INI
- Settings for the Online-Backup, 1134

Protect
- DFB, 776
- Project, 776

Q

Quantum
- Memory optimization, 147, 161

Quantum example
INTERBUS control, 919
Profibus DP controller, 933
Quantum-Peer Cop, 949
Remote control with DIO, 909
Remote control with RIO, 889
Remote control with RIO (series 800), 897
SY/MAX controller, 925
Quantum first startup
DOS Loader, 1060, 1094
EXECLoader, 1044, 1074
Modbus, 1044, 1060
Modbus Plus, 1074, 1094
Quantum Security Parameters, 133

R

Range Monitoring
ARRAY, 583
RDE, 587
converting RDE templates, 590
Cyclical Setting of Variables, 594
General, 588
Settings in the INI-File, 1129
RDE editor
toolbar, 836
reactivate flash save, 657
Reading the global DFBs
Settings in the INI-File, 1121
Reference Data Editor
Cyclical Setting of Variables, 594
General, 588
Reference data editor, 587
Changing signal states of a Located variable, 592
Replacing variable names, 602
reference data editor
converting RDE templates, 590
Reference Offset
LL984, 448
Reference Zoom
LL984, 451
References
LL984, 446
Register Address Format
INI File Settings, 1120
Remote controller with DIO, 914
Remote controller with RIO, 894
Remote controller with RIO (series 800), 903
Remote MBX Driver
Modbus Plus, 1033
replace
coil, LD, 245
contact, LD, 245
FFB, FBD, 211
FFB, LD, 245
Replace References
LL984, 453
Replacing variable names
Reference data editor, 602
Requirements
LL984, 444
RTU extension
Compact configuration, 127
Configure, 127
Runtime Error
I/O Event Sections, 1156

S

Save To Flash, 654
Saving the global DFBs during Upload
Settings in the INI-File, 1121
Scan
Constant, 649
Scan Rate
Timer Event Sections, 1141
scan times
single, 650
Search and Replace
Variable names and addresses, 540
searching and pasting
variable names and addresses, 544
searching and pasting variable names and addresses, 544

Section, 47
  Animation, 679
  Disable, 49
  Execution order, 48
  Export, 693
  Import, 699, 705, 714, 715, 716
  import, 700
  LL984, 441
  Status, 810
section
  import, 711
Secure Application, 19
Security, 763, 764, 766, 774, 775, 776
Segement
  LL984, 441
Segment manager, 99
Select
  LL984, 449
Select process information
  General information, 662
  Status and memory, 661
Selecting process information
  Status and memory, 661
separators
  data type editor, 574
  derived data type, 574
Set/Change PLC Password, 658
Setting up and controlling the PLC, 647
setup and control PLC
  general information, 648
SFC
  'SFCSTEP_STATE' variable, 264
  'SFCSTEP_TIMES' variable, 263
  Action, 265
  Action variable, 265
  Actions, 285
  alternative branch, 273
  Alternative connection, 275
  animation, 298, 300
  Calling up macros, 529
  Edit, 280
  edit, 279
  Icon bar, 830
  Identifier, 288
  initial step, 262
  Jump, 272
SFC
  Learn monitoring times, 303
  Link, 271
  maximum supervision time, 262
  minimum supervision time, 263
  Objects, 261
  Online, 297
  online functions, 298, 300
  Parallel branch, 276
  Parallel connection, 277
  Short Cut Keys, 843
  Step, 262
  step delay time, 262
  step duration, 262
  Step properties, 283
  string, 300
  Text object, 278
  transition, 267, 291
  Transition diagnosis, 306
  Transition section, 269
  Transition variable, 270
  waiting step, 262
Short Cut Keys, 840, 841, 843, 854
Short cut keys, 825, 839
Shortcut keys, 847
Simple sequences, 271
Simulate
  SPS, 753, 755
Simulation, 751, 753, 755
single sweeps, 650
special options, 114
Specific data transfer
  Peer Cop, 955
Speed optimized LL984- Processing, 653
SPS
  Simulate, 753, 755
ST, 377
  Animation, 370
  Assign instructions, 407
  Block call up, 425
  Code generation, 433
  Expressions, 381
  Instructions, 408
  List of Symbols, 833
  Online functions, 370, 435

ST, 377
- Operands, 382
- Operators, 387
- operators, 384
- Program creation, 437
- Short Cut Keys, 841
- syntax check, 432

ST Command
- , 401, 402, 403
- &, 404
- =, 400
- AND, 404
- Boolean AND, 404
- Boolean OR, 405
- ELSE, 414
- ELSIF...THEN, 415
- Equal to, 400
- Less than, 402
- Less than or equal to, 403
- Not equal to, 401
- OR, 405
- XOR, 406

ST command
- -, 391, 397
- (), 388
- *, 393
- **, 390
- +, 396
- >, 398
- >=, 399
- Addition, 396
- Assignment, 409
- Call function block, 426
- CASE...OF...END_CASE, 416
- Complement formation, 392
- Declaration, 411
- Division, 394
- Empty instruction, 423
- EXIT, 422
- Exponentiation, 390
- FOR...TO...BY...DO...END_FOR, 417
- FUNCNAME, 389
- function invocation, 430

ST command
- Greater than, 398
- Greater than/Equal to, 399
- IF...THEN...END_IF, 413
- MOD, 395
- Modulo, 395
- Multiplication, 393
- Negation, 391
- NOT, 392
- REPEAT...UNTIL...END_REPEAT, 421
- Subtraction, 397
- Use of parentheses, 388
- VAR...END_VAR, 411
- WHILE...DO...END_WHILE, 420

ST commandl
- /, 394

ST Comment
- Comment, 424

Start behavior
- Variable, 44

Start behavior of digital outputs, 46

Startup
- Presettings for Modbus, 1037
- Presettings for Modbus Plus, 1023

Startup with DOS Loader
- Modbus, 1059
- Modbus Plus, 1093

Startup with the EXECLoader
- Modbus, 1043
- Modbus Plus, 1073

State of the PLC, 646

Status, 632

Status bar, 810

ST-Command
- Boolean Exclusive OR, 406

step, 262
- alias designations, 293

step delay time, 262

step duration, 262

Step properties
- Process, 283

string
- control, 300

Structure
  PLC Memory, 139
  Program, 35, 36
  Project, 35, 36
Structured text, 377
Structured variables
  Import, 723
Subroutines
  LL984, 455
Symax-Ethernet
  specify coupling modules, 107
Symbols, 825, 833, 834
Syntax
  Data type editor, 567
  Derived Data Type, 567
syntax check
  IL, 366
  ST, 432

T

Tag
  IL, 321
TCP/IP
  Network Configuration, 987
TCP/IP network link, 644
TCP/IP-Ethernet
  specify coupling modules, 107
Text Object
  FBD, 207
Text object
  LD, 242
  SFC, 278
Timer Event Section
  Handling, 1135
Timer Event Sections, 1140
  Define Scan Rate, 1141
  Defining the Phase, 1142
  Examples for Parameterization, 1149
  Execution Order, 1145
  Operating System, 1146
Tool bar, 827, 828, 829, 830, 832
Toolbar, 837
toolbar, 836
Tools, 21
  Settings in the INI file, 1130
Trace
  LL984, 453
Transfer problems
  Modbus Presettings, 1042
transition, 267
  alias designations, 293
  declare, 291
  process, 291
Transition diagnosis, 306
Transition section, 269
Transition variable, 270

U

UDEFB
  FBD, 203
  LD, 237
Unconditional Configuration, 87
unconditional configuration
  precondition, 88
Unconditional locking of a section, 597
Undo
  LL984, 449
Upload PLC, 674
Uploading the PLC, 674
User-defined Elementary Function
  FBD, 203
user-defined elementary function
  LD, 237
User-defined Elementary Function Block
  FBD, 203
user-defined elementary function Block
  LD, 237
Utility program, 21

V

Variable
  Export, 697
  Start behavior, 44
Variable Editor
  Declaration, 537
Variable editor
  Exporting located variables, 547
  Search and Replace, 540

variable editor
searching and pasting, 544
Variable Storage
INI File Settings, 1120
Variables, 42
ASCII message editor, 610
Import, 719, 723, 726
LL984, 446
variables
import, 720
Variables editor, 535
Variables-Editor
General, 536
VARINOUT variables, 478
Various PLC settings, 63
View Tool, 685
Virtual MBX Driver
Modbus Plus, 1031

W

waiting step, 262
Warm restart, 44
Window, 807
Window elements, 810
window types, 808
Windows, 805
Check box, 816
Command buttons, 816
Dialog boxes, 815
Lists, 816
Menu commands, 813
Option buttons, 816
Status bar, 810
Text boxes, 816
Window, 807
Window elements, 810
windows
window types, 808